CAD-based geometry parametrisation for shape optimisation using Non-uniform Rational B-splines

Xingchen Zhang

School of Engineering and Materials Science

Queen Mary University of London

A thesis submitted for the degree of

Doctor of Philosophy

September 2017

This thesis is dedicated to my family:

wife: Yifang Zhang

parents: Wenhua Zhang, Tianxiu Sun

brother: Yang Zhang

for your love

Statement of originality

I, Xingchen Zhang, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature: Xingchen Zhang

Date: 11/09/2017

Abstract

With the continuous growth in computing power, numerical optimisation is increasingly applied in shape optimisation using Computational Fluid Dynamics (CFD). Since CFD computations are expensive, gradient-based optimisation is preferable when the number of design variables is large. In particular the recent progress with adjoint solvers is important, as these solvers allow to compute the gradients at constant computational cost regardless of the number of design variables, and as a consequence enable the use of automatically derived and rich design spaces.

One of the crucial steps in shape optimisation is the parametrisation of the geometry, which directly determines the design space and thus the final results. This thesis focuses on CAD-based parametrisations with the CAD model continuously updated in the design loop. An existing approach that automatically derives a parametrisation from the control points of a net of B-Spline patches is extended to include NURBS. Continuity constraints for water-tightness, tangency and curvature across patch interfaces are evaluated numerically and a basis for the resulting design space is computed using Singular Value Decomposition (SVD).

A CAD-based shape optimisation framework is developed, coupling a flow solver, an adjoint solver, the in-house CAD kernel and a gradient-based optimiser. The flow sensitivities provided by the adjoint solver and the geometric sensitivities computed through automatic differentiation (AD) are assembled and provided to the optimiser. An extension to maintain the design space and hence enables use of a quasi-Newton method such as the BFGS algorithm is also presented and the convergence improvements are demonstrated.

The framework is applied to three shape optimisation cases to show its effectiveness. The performance is assessed and analysed. The effect of parameters that can be chosen by the user are analysed over a range of cases and best practice choices are identified.

Acknowledgments

The past four years at Queen Mary University of London have been a memorable journey in my life. I would like to take this opportunity to thank many people who helped me directly or indirectly during this journey.

First of all, I would like to thank my supervisor, Dr. Jens-Dominik Müller, for giving me the opportunity to study under his guidance. This thesis would not have been possible without his help. His passion for research and wide knowledge are constant supports. He always provides valuable insights and suggestions to improve this work.

I am thankful for Prof Tom Verstraete, who is an associate professor at the von Karman Institude for Fluid Dynamics and a visiting professor at QMUL in the past two years, for his insightful suggestions and guidances, as well as encouragements.

I would also like to thank my past and present colleagues at the CFD optimisation group. The supports I received from Siamak Akbarzadeh, Rejish Jesudasan, Mateusz Gugala, Jan Hückelheim, Shenren Xu, Yang Wang, Pavanakumar Mohanamuraly and Orest Mykhaskiv are especially appreciated.

I also want to thank my examiners, Prof. Manolis Gavaises and Prof. Harvey Thompson, who gave very insightful comments to help me to greatly improve the quality of this thesis.

This work is funded by the China Scholarship Council [No. 201306230097] and Queen Mary University of London. Supports from the European project About Flow and IODA coordinated by Dr. Jens-Dominik Müller are also gratefully appreciated.

Special thanks have to go to my parents and brother, who always have faith in me and give me encouragements. I am lucky to have you as my family members.

Finally, I would like to express my sincere gratitude to my beloved wife, Yifang Zhang, who is my pretty sunshine. She makes me feel a better and wonderful world. Without the unparalleled encouragement, love and support she gives to me, this thesis would not have been possible.

Table of Contents

List of Figures

List of Tables

List of Symbols

$\mathbf{A}$	Jacobian
$\mathbf{B}$	inverse of the Hessian matrix
$\mathbf{C}$	the constraint matrix
C_0	parametric continuity C_0
C_1	parametric continuity C_1
$\mathbf{C}_0$	the constraint matrix which only contain G_0 entries
$\mathbf{C}_1$	the constraint matrix which only contain G_1 entries
D	drag force
e	internal energy
$\mathbf{e}_i$	unit vector in direction i
f	a function of the independent variable x
$\mathbf{f}$	the change in residual with respect to changes in shape
f_T	inflation factor used in the estimate of test point number
$\mathbf{f}_c$	convective flux vector
$\mathbf{f}_v$	viscous flux vector
$\mathbf{g}_k$	gradient of cost function in iteration k
G_0	geometry continuity G_0
G_1	geometry continuity G_1
h	step size in finite differences
$\mathbf{H}$	Hessian matrix
H	a projective mapping
j	the imaginary unit
J	cost function
$\mathbf{J}_i$	Jacobian matrix in point inversion algorithm
k	thermal conductivity coefficient
$\mathbf{K}$	a unitary matrix in singular value decomposition
$\mathrm{Ker}(\mathbf{C})$	the nullspace of the constraint matrix $\mathbf{C}$
$\mathrm{Ker}(\mathbf{C}_0)$	the nullspace of $\mathbf{C}_0$
L	lift force

$\mathbf{L}$	the diagonal scaling matrix
$\mathbf{n}$	surface normal direction
$\mathbf{n}_L$	the unit normal vector of the tangent plane on the left side
$\mathbf{n}_R$	the unit normal vector of the tangent plane on the right side
N_i	number of non-zero knot intervals in a knot vector
N_k	number of knots in a knot vector
$N_{i,p}$	p-th degree B-spline basis functions
$N_{j,q}$	q-th degree B-spline basis functions
N_M	number of zero knot intervals because of internal multiplicities in a knot vector
N_p	number of control points of a curve
N_q	degree of a spline in computing the number of test points
p	pressure
$\mathbf{p}_k$	search direction in the k-th optimisation iteration
$\mathbf{P}$	control points coordinates
$\mathbf{P}_{i,j}$	control point coordinates of index (i, j)
$\mathbf{P}^\omega$	control points in homogeneous form
$\mathbf{P}^s$	scaled control points
$\mathbf{q}$	source term vector in RANS equations
r	theoretical rank of a matrix
r'	numerical rank of a matrix
$\mathbf{R}$	conservative residual of the discretised flow equations
$R_{i,j}$	rational basis functions
s_k	step size in the k-th optimisation iteration
S	cross section area
$\mathbf{S}$	NURBS or B-spline surface
$\mathbf{S}^\omega$	NURBS surface in homogeneous form
S_{M_x}	momentum source in x direction
S_{M_y}	momentum source in y direction
S_{M_y}	momentum source in y direction
$\mathbf{S}_u$	the derivative of NURBS surface w.r.t. the parametric coordinate: $\frac{\partial \mathbf{X}_s}{\partial u}$
$\mathbf{S}_v$	the derivative of NURBS surface w.r.t. the parametric coordinate: $\frac{\partial \mathbf{X}_s}{\partial v}$
$\mathbf{S}_{uu}$	the derivative of NURBS surface w.r.t. the parametric coordinate: $\frac{\partial^2 \mathbf{X}_s}{\partial u^2}$
$\mathbf{S}_{vv}$	the derivative of NURBS surface w.r.t. the parametric coordinate: $\frac{\partial^2 \mathbf{X}_s}{\partial v^2}$
$\mathbf{S}_{uv}$	the derivative of NURBS surface w.r.t. the parametric coordinate: $\frac{\partial^2 \mathbf{X}_s}{\partial u \partial v}$
S_ϕ	the source term of ϕ in transport equation
t	time
$\mathbf{u}$	perturbation field
$\mathbf{u}_v$	velocity vector
$\mathbf{U}$	state variables

$\mathbf{v}$	the adjoint variable
$\mathbf{V}$	a unitary matrix in singular value decomposition
$\mathbf{W}$	conservative variable vector in RANS equations
x	independent variable
$\mathbf{X}_s$	surface mesh coordinates
$\mathbf{X}_v$	volume mesh point coordinates / spatial discretisation
$\boldsymbol{\alpha}$	a vector of design variables
$\boldsymbol{\alpha}_k$	a vector of design variables in iteration k
α_i	one particular design variable
Γ	diffusivity coefficient
δG_1	the deviation value from exact G_1
$\delta \mathbf{P}_r$	the recovery displacement of control points
ϵ	step size in complex step method
ϵ_1	a measure of Euclidean distance in point inversion problem
ϵ_2	a zero cosine measure in point inversion problem
η_0	a small positive constant in the first Wolfe condition
η_1	a positive constant in the second Wolfe condition
η_2	a positive constant in the third Wolfe condition
$\boldsymbol{\kappa}_i$	vector of function values in point inversion algorithm
μ	dynamic viscosity
σ_C	the cut-off threshold value in singular value decomposition
ρ	density
ϕ	passive scalar
Φ	dissipation function
ω	weight of control points
$\omega_{i,j}$	weights of control point of index (i, j)
Ω	volume of control volume

List of Abbreviations

AboutFlow	Adjoint-based Optimisation of Industrial and Unsteady Flows
AD	Automatic Differentiation or Algorithmic Differentiation
ADODG	AIAA Aerodynamic Design Optimisation Discuss Group
AoA	Angle of Attack
BFGS	Broyden-Fletcher-Goldfrab-Shanno
BI-CGSTAB	Bi-Conjugate Gradient Stabilized Method
BRep	Boundary Representation
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CAPRI	Computational Analysis PRogramming Interface
CDS	Central Difference Scheme
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy
CGSTAB	Conjugated Gradient Stabilized Method
CP	Control Point
CV	Control Volume
DoF	Degree of Freedom
DLR	German Aerospace Centre
EA	Evolutionary Algorithms
FD	Finite Differences
FDM	Finite Differences Method
FEM	Finite Element Method
FFD	Free Form Deformation
FlowHead	Fluid Optimisation Workflows for Highly Effective Automotive Development Processes
FVM	Finite Volume Method
GA	Genetic Algorithms
GUI	Graphic User Interface

IGA	Isogeometric Analysis
IGES	Initial Grahpics Exchange Specification
IODA	Industrial Optimal Design using Adjoint CFD
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
KKT	Karush–Kuhn–Tucker
LAPACK	Linear Algebra Package
MDO	Multidisciplinary Design Optimisation
NOED	International Conference on Numerical Optimisation
N-S	Navier-Stokes
NSPCC	NURBS-based Parameterisation with Continuity Constraints
NURBS	Non-Uniform Rational B-Splines
OCCT	Open Cascade Technology
ONERA	Office National d'Etudes et Recherches Aérospatiales
O-O	Operator Overloading
PARSEC	Parameterised Sections
PCA	Principal Component Analysis
PIV	Particle Image Velocimetry
POD	Proper Orthogonal Decomposition
QMUL	Queen Mary University of London
QUB	Queen's University of Belfast
QUICK	Quadratic Upstream Interpolation for Convective Kinematics
RANS	Reynolds-Averaged Navier-Stokes
SA	Spalart-Allmaras
S-T	Source Code Transformation
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
STAMPS	Source-Transformation Adjoint Multi-Physics Solver
STEP	Standard for the Exchange of Product Model Data
STL	surfaces with triangula-tions
SVD	Singular Value Decomposition
UDS	Upwind Differencing Scheme
VKI	von Karman Insitute

Chapter 1

Introduction

1.1 Background

The design of objects such as aircraft [1, 2], turbine blades [3, 4], car bodies [5–7] and air ducts [8, 9] is very important in industry and beneficial to human life. The aerodynamic performance and efficiency of these objects are significantly determined by their shape. Thus, proper shape design is essential for these products.

In the past, the design of these shapes could be time consuming and expensive. Normally, prototypes are needed to be built and tested repeatedly until a satisfactory design is obtained. The emergence of Computational Fluid Dynamics (CFD) provides an alternative to the experimental facilities, which allows the designs to be tested immediately and requires short time varying from a few hours to a few days. The design process is still iterative though, namely CFD simulations need to be run again and again based on newly gained knowledge, to obtain a good design.

Another feature of previous shape design is that it relies heavily on the intuition and experience of engineers, as illustrated in Fig. 1.1. There are several problems with manual design. Firstly, to improve current designs, one has to optimise complex flow features which may not be intuitive to a human designer. Secondly, manual evaluation of the design and change of the parameters are required, which are labour-intensive and have high demand on the designer's expertise. This problem leads to that only small design spaces, namely a small number of design variables, are possible, thus the design spaces are not fully explored.

Nowadays, the engineering design is facing ever-increasing requirements in engineer-

ing design such as performance, environmental impact and lift-time cost. To satisfy these needs, large design spaces with a large number degrees of freedom (DoF) need to be explored systematically, which can only be achieved through numerical optimisation. Therefore, numerical optimisation algorithms have attracted widespread interests in recent years and are increasingly used in advanced design.

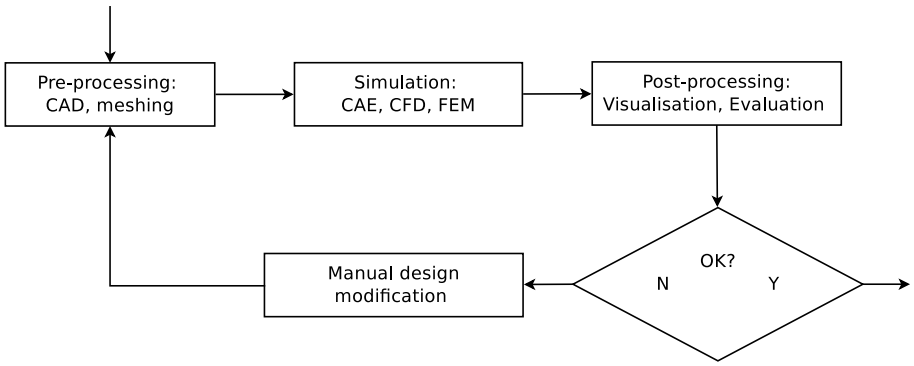


Figure 1.1: Manual design loop.

The design loop of numerical optimisation is shown in Fig. 1.2. The key element of this loop is the optimiser, which replaces the human designer in Fig. 1.1. Early studies presented by Hicks et al. [10, 11] demonstrated the feasibility of applying numerical optimisation in airfoil and wing shape design. Since then, many efforts have been put in this direction.

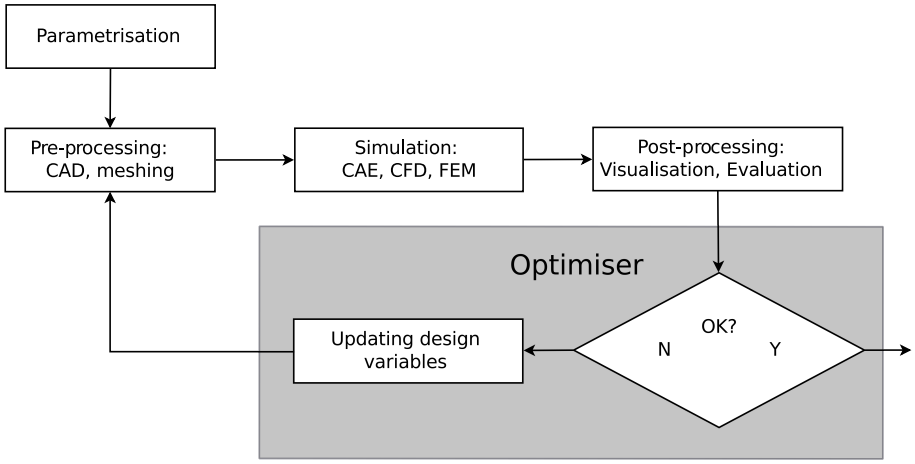


Figure 1.2: Design loop with numerical optimisation.

One of the crucial steps in shape optimisation is parametrisation. The parametrisation of geometry is generally about choosing method to describe it, while in shape optimisa-

tion, parametrisation specifically means the process of choosing design variables. Indeed, parametrisation transforms the engineering problem into a mathematical one thus it determines the design space and has significant impact on optimisation results. Consequently, the choice of proper parametrisation method is crucial in shape optimisation problem.

For parametrisation, an expected property is to exchange data among different systems, for instance between optimisation chain and manufacturing system. This is especially important in multi-disciplinary design. To this end, the Computer-Aided Design (CAD)-based parametrisation [12] stands out as a suitable choice for us. Because in industry, the CAD model is needed to manufacture an object[1]. This work focuses on the CAD-based parametrisation method in gradient-based aerodynamic shape optimisation.

1.2 CAD-based aerodynamic shape optimisation

A general CAD-based aerodynamic shape optimisation framework is shown in Fig. 1.3. Essential components of the framework include a CFD solver, an optimiser, an adjoint solver (flow sensitivity calculation) and the parametrisation. This figure shows clearly the working mechanism of the framework, and also how different components interact with each other. To be more specific, a CAD engine is used to read and write geometries, and compute CAD sensitivities (see Section 5.4.6). A CFD solver (see Chapter 4) is used to perform flow analysis, based on which the adjoint solver is derived to compute flow sensitivities. These two parts of sensitivities are then assembled and provided to the gradient-based optimiser to find a direction along which the design can be improved.

This section will give a brief introduction to each ingredient, and more details will be given in later chapters.

1.2.1 Computational fluid dynamics

CFD is the analysis of systems involving fluid flow, heat transfer and associated phenomena such as chemical reactions by means of numerical simulation [13]. It is regarded as the 'virtual wind-tunnel' [14], and the past 20 years have witnessed a major increase in its use due to the advantages in following aspects: turnaround time, exact similarity, non intrusive, analysis, feasibility and extendibility [14].

CFD simulations have been established in industry and academia as a tool to predict

[1] http://www.vista-industrial.com/blog/the-importance-of-cad-models-in-manufacturing/

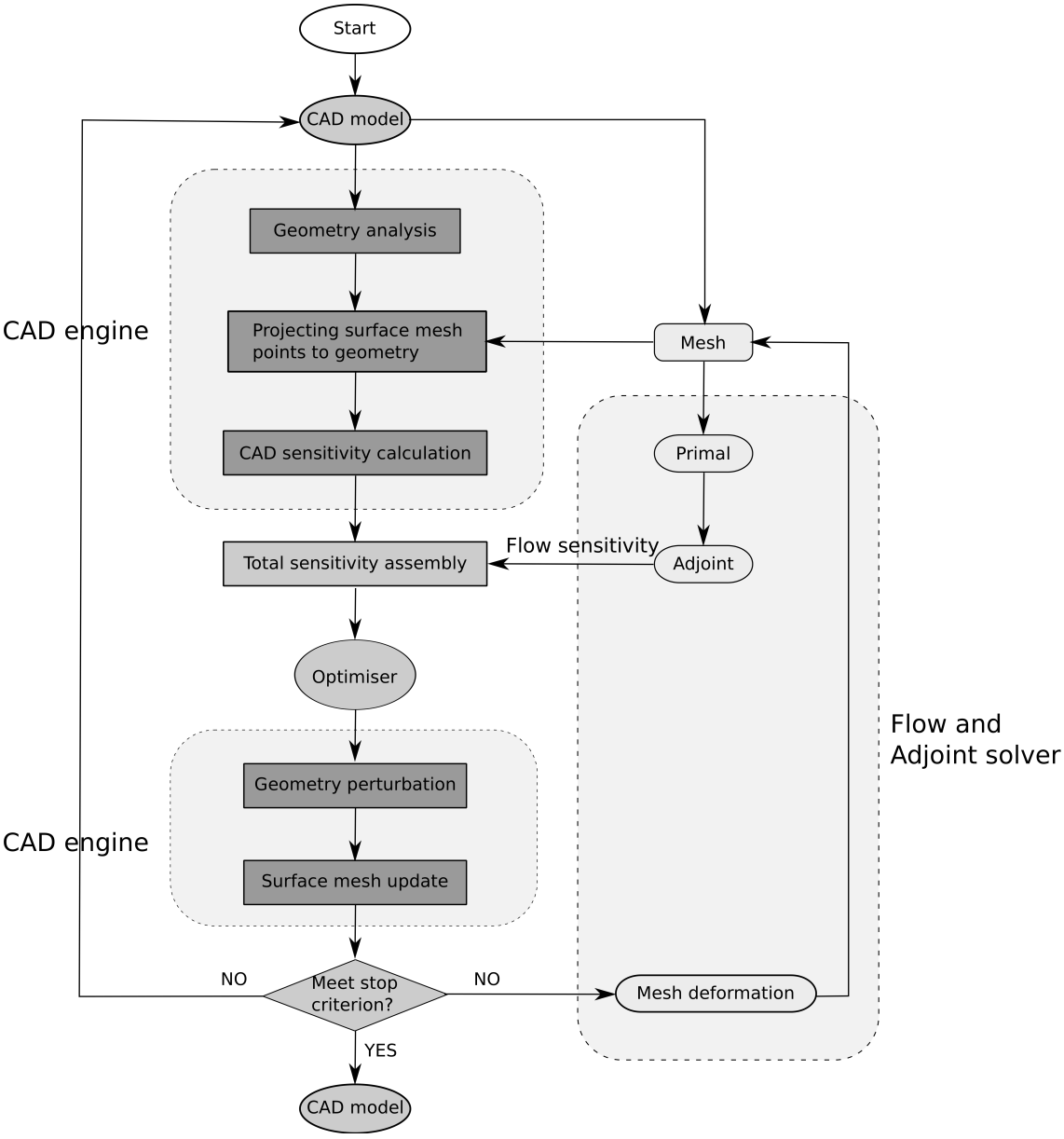


Figure 1.3: The schematic overview of a CAD-based CFD shape optimisation framework.

the behaviour of flows and have been widely used in aerodynamic shape optimisation [13, 15]. Owing to the continuous progress of the computational power and algorithms, aerodynamic analysis with CFD are now routinely performed by using Navier-Stokes solver in many cases, such as the three-dimensional wing configurations [16–18] and cooling channel of turbine blade [19]. Indeed, most industrial projects have replaced large part of experiments in the design process with CFD. Experiments are only carried out in the final design stage to validate the CFD results.

In the aerodynamic shape optimisation framework, CFD is used to perform flow analysis. In addition, the adjoint solver (see Chapter 4) is also derived based on flow solver to provide computational inexpensive gradients.

1.2.2 Numerical optimisation

With numerical optimisation, the optimiser selects a new set of values for design variables and the design spaces can be systematically explored. It has been applied in many areas, such as wing design [11, 12, 20–22], optimal design of turbo-machinery [23–27], structural optimisation of solids [28–30].

Generally speaking, there are two kinds of optimisation methods: gradient-free and gradient-based methods. Gradient-free optimisation approaches, such as Genetic Algorithms (GA) and Evolutionary Algorithms (EA), are very well established, especially for linear structural optimisation where the computational cost of an evaluation is low. In these methods, only the value of the objective function is needed at each optimisation step. However, these methods require a large number of function evaluations when going beyond 50-100 design variables. For example, the GA may require 5 to 200 times of function calls of gradient-based method [31]. The cost of gradient-free methods then becomes prohibitive when used with expensive CFD models, since in many cases a single CFD evaluation could last for a long time, ranging from a few hours to several days [18]. Therefore, gradient-free methods are normally used in cases where the number of design variables is small, such as the optimisation of car body shape [32, 33]. On the other hand, in shape and topology optimisation with CFD, gradient-based methods are preferred because the convergence of them suffers much less from large design spaces. In other words, gradient-based approaches can converge quickly to the optimum with fewer evaluations, compared to their gradient-free counterparts.

1.2.3 Sensitivity calculation

The gradient-based methods require gradient information of the objective function, as they point out the direction where the design can be improved. This replaces the hurdle of computational cost with the challenge to compute the gradients for all components in the simulation chain, i.e. not only the flow solver, but also the parametrisation. Thus, calculating the gradient efficiently and accurately is crucial.

Gradient computation can be achieved with several methods, and all of them have their own advantages and disadvantages. Finite differences (FD) is an easily implemented

method, however it could incur truncation or round-off errors that can be difficult to control. Also, the computational cost scales with the number of design variables. Alternatives are the complex step method [34] or algorithmic differentiation (AD) [35, 36], which can produce exact derivatives, but the computational cost are same as FD. Another one is the adjoint method, which allows to compute the gradients of an objective with respect to an arbitrary number of design variables at constant cost and has hence become the method of choice in CFD [37, 38]. However, implementing an adjoint solver is not trivial, both the continuous approach [39] and the discrete one [40–42] require significant effort.

In this work, the CFD solvers use the reverse mode of AD to produce a discrete adjoint solver, while forward mode AD is utilised to compute the geometry sensitivities. Details on CFD solver and adjoint will be given in Chapter 4, and the gradient computation will be discussed in Section 3.4 in detail.

1.2.4 Parametrisation

There are various parametrisation approaches and each of them has its own pros and cons. All the parametrisation methods (see Chapter 2) can be and have been integrated into the design loop, except full CAD-based system. However, the typical CAD-free parametrisation approaches, such as free-form deformation (FFD) [43] or node-based deformation [44], produce the optimal shape as a deformed mesh which needs to be translated back to CAD manually. In this process, fine details of the shape variation could get lost, which impairs the optimal performance. For general industrial application, a parametrisation in CAD is needed that integrates the geometry description in the CAD-CAM (Computer-Aided Manufacturing) virtual prototyping chain. Closing this gap by including the CAD in the design loop will be a significant step forward for general application of numerical optimisation in routine industrial design.

On the other hand, the parametrisation method also affects the choice of sensitivity calculation approach. For example, the node-based parametrisation (see Section 2.3.1.1) leads to numerous design variables, making the computational cost of gradient computation unacceptable unless the adjoint method is utilised.

A main task of this research is to investigate CAD-based parametrisation method. A more detailed introduction to parametrisation, as well as an extensive literature review of various parametrisation methods will be presented in Chapter 2.

1.3 Motivation

It is important to have a CAD format as the result of shape optimisation, for convenience of further post-processing and manufacturing. Therefore, the CAD system should be included in the design loop as shown in Fig. 1.3. In gradient-based optimisation, sensitivity is needed. However, because the source codes of commercial CAD systems are not available, only FD can be used to compute CAD sensitivities. As a consequence, it seems that it is not suitable to include commercial CAD systems in the design loop.

One alternative to obtain CAD sensitivities is to differentiate a full open source CAD system, such as OpenCASCADE[2], using AD. In this way, the parametrisation is explicitly defined in the CAD system (see Section 2.3.2). Some of the author's colleagues at Queen Mary University of London (QMUL) are following this paradigm [45]. Another option is to write a reduced CAD kernel based on the boundary representation (BRep), from which the parametrisation is automatically derived (see Section 2.3.2). Then, differentiate this CAD kernel using AD. The present work follows this way.

BRep is the most commonly used approach to define a geometry in CAD system [46], such as CATIA[3] and Solidworks[4]. The main advantage of deriving a parametrisation from the BRep is that it is generic and can be interpreted independent of CAD systems, i.e. it can be used to exchange data between different systems. In BRep, the skin of the geometry is defined analytically by surface patches which are stitched together. BRep models consist of two parts: topology (faces, edges and vertices) and geometry (surfaces, curves and points). Common formats of BRep model are the Initial Graphics Exchange Specification (IGES) and Standard for the Exchange of Product Model Data (STEP) format, one example of the topology information and corresponding geometry information in STEP file is illustrated in Fig. 1.4. The STEP file will be introduced in Section 5.8.

The CFD optimisation group at QMUL has developed a novel CAD-based parametrisation approach over several years based on BRep, which efficiently integrates the CAD system inside the gradient-based design loop and robustly produces a CAD shape as the output. The idea was firstly proposed by Yu et al. [47], where the BRep model of a 2-D geometry was utilised to provide geometry information in the form of STEP file. Then the approach was further developed by Xu et al. [48, 49] to deal with geometric continuity constraints between adjacent patches and termed NURBS-based parametrisation with continuity constraints (NSPCC). This outperforms those methods which only consider

[2] https://www.opencascade.com/
[3] https://www.3ds.com/products-services/catia/products/v5/portfolio/
[4] http://www.solidworks.co.uk/

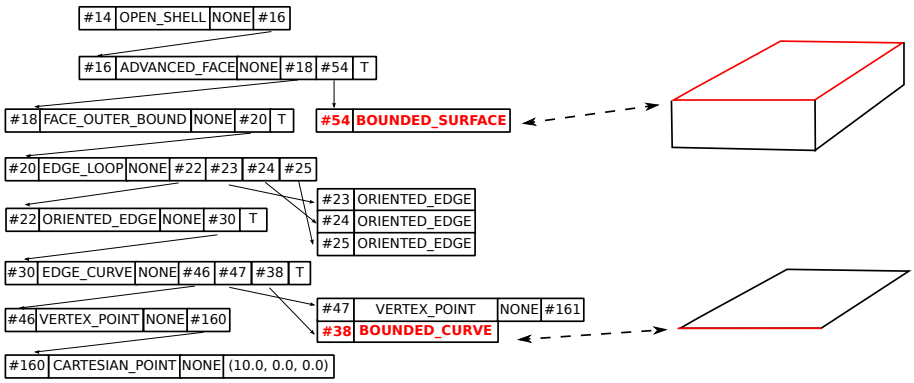


Figure 1.4: Topology and geometry information in BRep.

one single patch [50].

However, there are several things remaining unclear thus hindering the effectiveness of STEP-based parametrisation and optimisation. These problems are as following:

- Firstly, the Non-Uniform Rational B-Splines (NURBS) (see Section 2.2) are the de facto industry standard for the representation, design and data exchange of geometry informations [51], and thus are the basis of STEP file. However, most STEP-based methods including current implementation of NSPCC [48] are based on B-splines, although termed NURBS-based parametrisation. Compared to B-splines, NURBS support wider range of geometries, namely NURBS can accurately express conic and circular shape which are commonly used in engineering [52], like the half-cylinder shown in Fig. 1.5. While with B-splines, approximation is needed. More importantly, NURBS provide more degrees of freedom in controlling geometries, i.e. the rational weights (see Section 2.2). However, the benefits of NURBS in CAD-based aerodynamic shape optimisation are still not very clear. In addition, how to utilise NURBS effectively in shape optimisation is remained to be explored.

- Secondly, the main benefit of numerical optimisation is to systematiclly explore large design space. However, good performance of current NSPCC implementation is heavily dependent on experience of users and the set up is not quite straightforward, which negates the benefit of numerical optimisation. Reasons for this have to be found and researches must be conducted to make the parametrisation method perform well more automatically.

- Thirdly, in theory different optimisation methods should work in the optimisation framework. However, in previous work [48, 49] only the steepest descent method (see Section 3.3.1) were utilised, and problems are encountered when trying to use

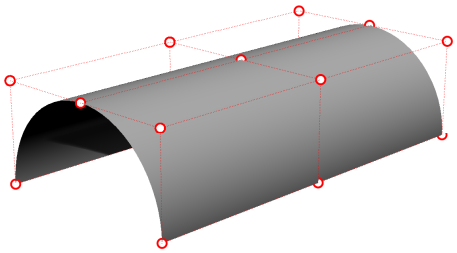


Figure 1.5: A half-cylinder geometry and its NURBS control points.

advanced optimiser. The reasons of this and measures to use advanced optimisers need to be investigated. This would make the optimisation framework more efficient.

The aim of this project is to investiage CAD-based parametrisation method and optimisation loop, and to solve aforementioned problems. This work is expected to contribute to the community by providing methods and informations to build a robust and powerful CAD tool, as well as an automated optimisation framework.

1.4 Thesis objectives

In light of above discussions, the primary objectives envisaged in this thesis are:

1. Investigate the benefits of using NURBS in aerodynamic shape optimisation, and explore how to utilise NURBS effectively. This can be further subdivided as follows:

   - Include weights attached to NURBS control points as additional design freedoms, and explore how to utilise weights effectively.
   - Demonstrate the advantage of NURBS compared to B-splines in shape optimisation problems.
   - Compute derivatives of NURBS surface using automatic differentiation (AD) and verify them.
   - Extend the in-house CAD kernel NSPCC to support NURBS. For example, it should support reading and writing NURBS geometries, such that wider range of geometries can be handled by NSPCC.

2. Further investigate the current NSPCC approach to add understanding, and develop NSPCC accordingly to make it more powerful, automatic and easy to use, includ-

ing:

- Investigate the constitution of design space, and factors affecting the optimisation performance, so as to find out how the design space is changed by parameters in the CAD kernel.
- Propose a pre-conditioner on the determination of effective design space.
- Demonstrate the ability of NSPCC to be coupled with different flow and adjoint solvers.
- Enrich the functionalities of NSPCC so that it can be applied to more shape optimisation problems.

3. Find out the reason why previously only the steepest descent method is used as optimiser, and explore strategy to couple advanced optimisation algorithms, such as quasi-Newton method, with NSPCC in the aerodynamic shape optimisation
4. Build an aerodynamic shape optimisation framework, which has following features:
   - Automated, i.e. no much human effort is needed during the whole optimisation process.
   - Use CAD format as input and output, which can be easily processed by CAD software.
   - Work with different solvers and optimisers.
5. Develop a method to impose thickness constraint in wing shape optimisation
6. Characterize the performance, effectiveness and efficiency of NSPCC and the CAD-based shape optimisation framework through applications to industrial cases.

1.5 Thesis outline

The outline of this thesis is as follows. From Chapter 2 to Chapter 4, the key basis for this work, which are also the key ingredients of shape optimisation framework, are discussed. Specifically, Chapter 2 introduces geometry parametrisation, including an introduction to NURBS and a literature review on parametrisation methods. Chapter 3 firstly introduces some basic knowledge on numerical optimisation, then summarises the commonly used methods of computing gradient. Chapter 4 introduces CFD and in-house solvers. Both flow and adjoint solvers used in this work are introduced briefly.

The following two chapters discuss the NURBS-based parametrisation method with geometric continuity (NSPCC). Chapter 5 introduces NSPCC in detail, especially the research on underlying principle and major developments made in this research. Chapter 6 presents validation results which make the approach more reliable.

From Chapter 7 to Chapter 9, the focus is the applications of the developed method in CFD-involved shape optimisation problems. Chapter 7 shows shape optimisation of an industrial 3-D S-bend air duct, where the capability of NPSCC to couple with incompressible solver is demonstrated. This chapter is also devoted to add understanding on the approach. Chapter 8 and Chapter 9 present the coupling of NSPCC with compressible solver. To be more specific, Chapter 8 presents the shape optimisation of a U-bend cooling channel from the turbomachine industry, which cannot be handled by previous version NSPCC. Chapter 9 investigates the aerodynamic shape optimisation of the ONERA M6 wing.

Finally, the conclusions, main contributions and future research recommendations are summarised in Chapter 10.

Chapter 2

Geometry parametrisation

Parametrisation is an essential ingredient in shape optimisation, as it has enormous impacts on the design space and thus the optimisation result. This chapter is devoted to introduce parametrisation approaches in detail and give an extensive literature review. Firstly, a general introduction to parametrisation is presented in Section 2.1, where requirements of a good parametrisation method are given. Secondly, the NURBS surface is discussed in Section 2.2. Then, detailed introduction and literature review on the CAD-free and CAD-based parametrisation methods are given in Section 2.3.1 and 2.3.2, respectively. Finally, Section 2.3.3 surveys the application of NURBS in shape optimisation.

2.1 Introduction to parametrisation

There exist a large variety of parametrisation methods. Samereh [53] classified those for high-fidelity multidisciplinary shape optimisation into eight categories: Basis vector, Domain element, Partial Differential Equation, Discrete (mesh point), Polynomial and Spline, Analytical, CAD-based and Free-Form deformation (FFD). Some methods are generic, for example, the discrete method can be applied to very wide applications. Some methods are however designed specifically for a particular kind of applications.

Actually, every method has its own advantages and drawbacks, one needs to choose a proper method according to requirements. Generally speaking, a good parametrisation method for routine industrial applications should hold the following features [48, 53]:

- Fully automatic, which does not need additional user input.

- Provide consistent geometry changes across all disciplines, and having a direct connection to the CAD system for design, analysis, post-processing and manufacturing.
- Can be integrated into the design loop at a reasonable computational cost.
- Have a rich design space of smooth deformations, namely rich enough to capture all relevant modes while maintaining smoothness.
- Support imposing constraints, such as manufacturing constraints and geometric continuity constraints.
- Provide sensitivities.

The data exchange among different procedures of the design and manufacturing process is normally achieved through CAD systems [54]. Based on whether CAD is involved or not, the parametrisation methods again can be distinguished broadly into two kinds, namely CAD-free and CAD-based methods. The former refers to those do not use CAD formats as input and can not return a CAD geometry after optimisation, while the CAD-based methods do. Figure 2.1 shows examples of CAD-free method and CAD-based parametrisation via NURBS (see Section 2.2), where the mesh nodes and control points of NURBS surface are used as design variables, respectively.

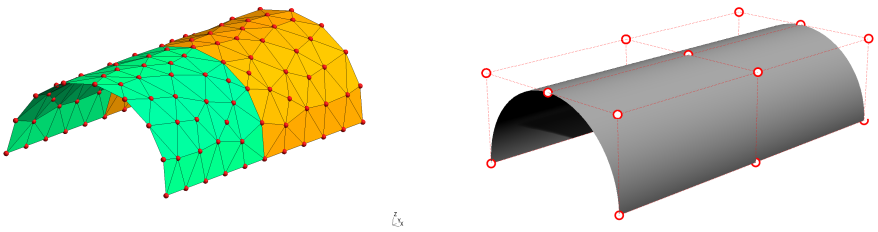


Figure 2.1: CAD-free approach and CAD-based parametrisation via NURBS. Left: surface mesh coordinates are used as design variables. Right: NURBS control points are used as design variables.

Note that depending on whether the link to CAD is available, the NURBS-based parametrisation approach can either be a CAD-based method, or a CAD-free one, as shown in Fig. 2.2. This thesis focuses on the CAD-based parametrisation via NURBS, which is a generic approach.

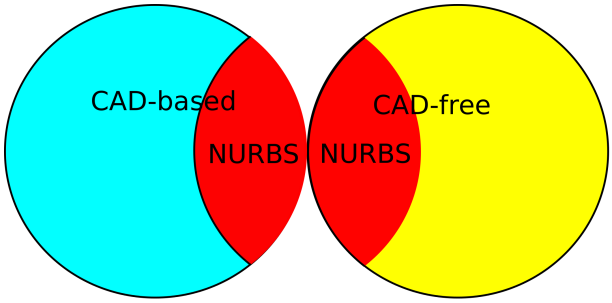


Figure 2.2: The category of parametrisation approaches.

2.2 Introduction to NURBS surface

Non-Uniform Rational B-splines (NURBS) are industry standard employed to represent and exchange geometric informations, on which many international standards, such as IGES and STEP, are based. One of the merits of using NURBS is that not only analytic shapes, such as conic sections and quadric surfaces, but also free-form geometries, such as car bodies and ship hulls, can be represented by a unified mathematical basis [51]. Therefore, a user can design surfaces without taking type of surfaces and special features into consideration [55, 56]. As a result, by incorporating the NURBS in the design loop, the effort to exchange information in a suitable format between different disciplines, such as aerodynamic/structural analysis and post-processing tools, is significantly reduced [57, 58].

NURBS are generalisation of B-splines and Bézier representations, thus the family of curves and surfaces that can be represented with NURBS is much wider [59]. A NURBS patch is a 3D surface defined as [51]:

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\omega_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\omega_{i,j}} \quad 0 \le u,v \le 1, \tag{2.1}$$

where $\mathbf{P}_{i,j}$ are control points, $\omega_{i,j}$ are the corresponding weights. $N_{i,p}(u)$ and $N_{j,q}(v)$ are p-th and q-th degree B-spline basis functions defined in the following knot vectors:

$$\{\underbrace{0,\ldots,0}_{p+1}, u_{p+1},\ldots,u_i,\ldots,u_{r-p-1},\underbrace{1,\ldots,1}_{p+1}\}$$
$$\{\underbrace{0,\ldots,0}_{q+1}, v_{q+1},\ldots,v_j,\ldots,v_{s-q-1},\underbrace{1,\ldots,1}_{q+1}\}$$

where $r = n+p+1$ and $s = m+q+1$. $N_{i,p}(u)$ and $N_{j,q}(v)$ are given by the following

recursive expression:

$$N_{i,0}(u) = \begin{cases} 1 & if \quad u_i \leq u < u_{i+1} \\ 0 & otherwise \end{cases}$$

$$N_{i,k}(u) = \frac{(u - u_i)}{u_{i+k} - u_i} N_{i,k-1}(u) + \frac{(u_{i+k+1} - u)}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u). \tag{2.2}$$

The basis functions are equal to zero everywhere except for an interval delimited by the order of NURBS, $[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$, defining the area of influence of each control point and its weight, as illustrated in Fig. 2.3. In these figures, curves from left to right are $N_{0,p}$, $N_{1,p}$, $N_{2,p}$, $N_{3,p}$, $N_{4,p}$, $N_{5,p}$, respectively. It can be observed that each basis function is non-zero only in an interval controlled by degree, namely higher degree leads to larger affection area. Also note the symmetry of the basis functions about the $u = 0.5$ line in Fig. 2.3.

Introducing the piecewise rational basis functions

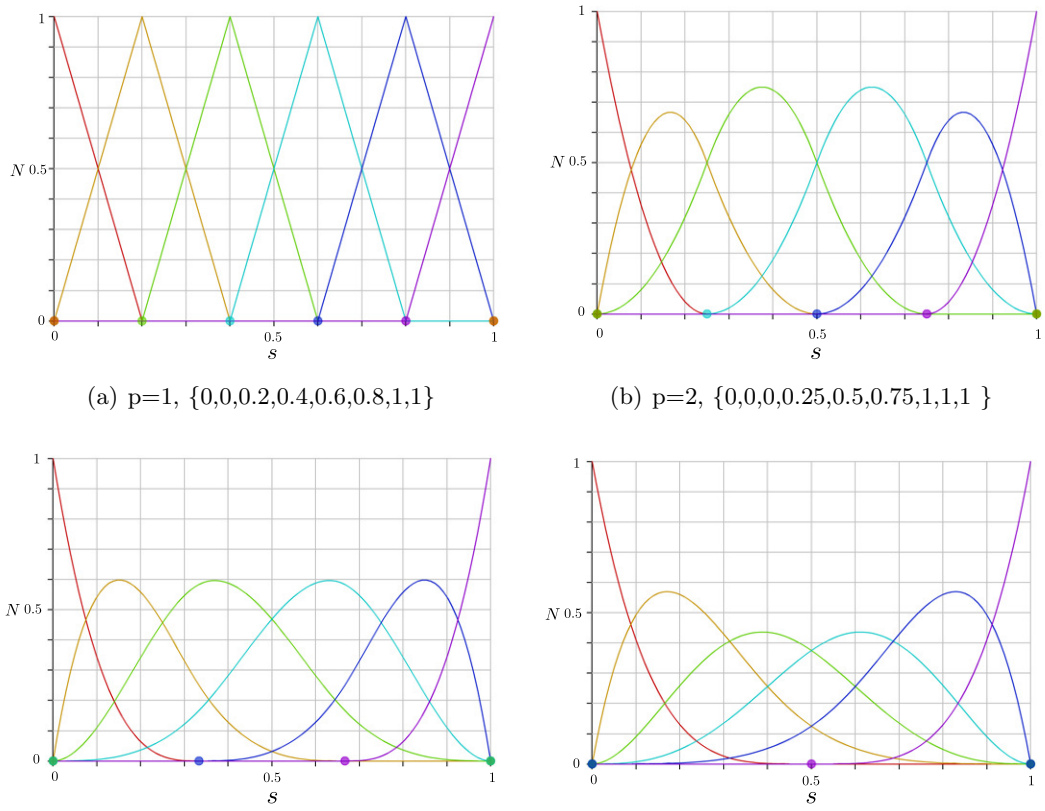
$$R_{i,j}(u, v) = \frac{N_{i,p}(u) N_{j,q}(v) \omega_{i,j}}{\sum_{k=0}^{n} \sum_{l=0}^{m} N_{i,p}(u) N_{j,q}(v) \omega_{k,l}}, \tag{2.3}$$

the surface equation (2.1) can be rewritten as

$$\mathbf{S}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j}(u, v) \mathbf{P}_{i,j}. \tag{2.4}$$

Piegl et al. [51] claimed the role of NURBS in CAD/CAM/CAE world is similar to the English language's role in science and business. This enormous popularity of NURBS is due to their important geometric properties which make then very powerful:

- NURBS are generalisation of Non-rational B-spline, Bézier and rational Bézier surfaces. This means by using NURBS, wider range of geometries can be described.
- Invariance. NURBS curves and surfaces are invariant under common geometric transformations, such as affine transformation (rotation, translation, scaling) and perspective projection [60]. For example, an affine transformation is applied to the surface by applying it to the control points. These properties make the using homogeneous form (see Section 5.2) possible.
- Local modification: due to the property of the basis functions in equation (2.2), if $\mathbf{P}_{i,j}$ is perturbed or its weight $\omega_{i,j}$ is changed, only the surface shape in the rectangle $[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$ will be affected. This property is important in aerodynamic shape optimisation, because the shape can be controlled locally.



(a) p=1, {0,0,0.2,0.4,0.6,0.8,1,1}

(b) p=2, {0,0,0,0.25,0.5,0.75,1,1,1 }

(c) p=3, {0,0,0,0,0.33,0.67,1,1,1,1}

(d) p=4, {0,0,0,0,0,0.5,1,1,1,1,1}

Figure 2.3: Basis function of different degree.

- Strong convex hull property: assume $\omega_{i,j} \geq 0$ for all control points. If $(u, v) \in [u_{i_0}, u_{i_0}+1) \times [v_{j_0}, v_{j_0}+1)$, then $\mathbf{S}(u, v)$ is in the convex hull of the control points $\mathbf{P}_{i,j}, i_0 - p \leqslant i \leqslant i_0$ and $j_0 - q \leqslant j \leqslant j_0$. This property may help to define geometric constraints.

- Differentiability: $\mathbf{S}(u, v)$ is $p - k$ times differentiable with respect to u at a u knot of multiplicity k; $\mathbf{S}(u, v)$ is $q - k$ times differentiable with respect to v at a v knot of multiplicity k. This property is related to the continuity of NURBS surfaces at knots.

Compared to B-splines, NURBS provide weight attached to each control point, thus having more freedoms in controlling the shape. Figure 2.4 presents a NURBS curve with 7 control points. Different weights are set to the control points $\mathbf{P}_4$, which clearly show how the weight affects the curve. Put specifically, a larger weight pulls the curve closer to this control point, while a smaller weight pushes the curve away from this control point.

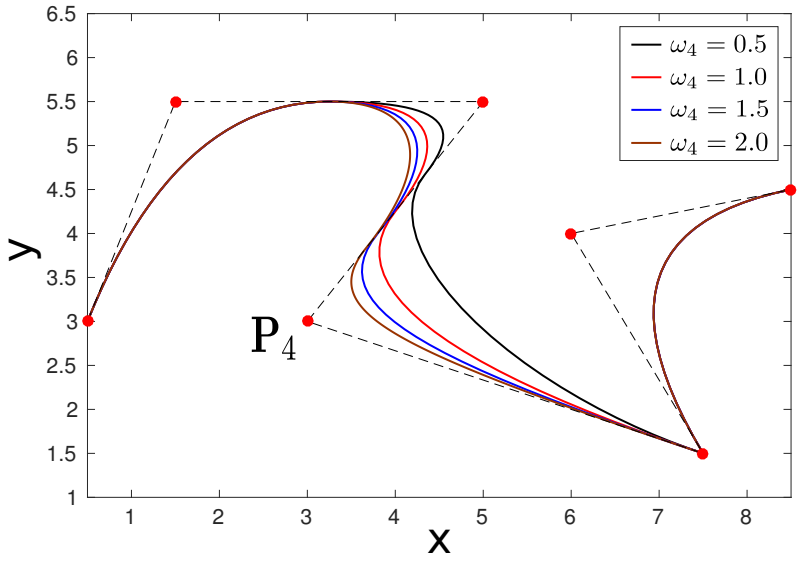


Figure 2.4: The effect of control point weight on NURBS curve.

When implementing NURBS, it is often preferable to represent NURBS in 4-D space, termed homogeneous space [51]. When described using homogeneous coordinates, a NURBS in 3D space is a B-splines in 4D space, which is an idea to represent a rational curve (or surface) in n-dimensional space as a polynomial curve (or surface) in $(n+1)$-dimensional space. A control point $\mathbf{P} = (x, y, z)$ in three-dimensional Euclidean space is embedded as $\mathbf{P}^\omega = (\omega x, \omega y, \omega x, \omega) = (X, Y, Z, W)$ in four-dimensional space, where $\omega \neq 0$ is the weight attached to this control point. $\mathbf{P}$ can be obtained from $\mathbf{P}^\omega$ by a perspective mapping H. In this mapping, the first three coordinates of $\mathbf{P}^\omega$ are divided by the fourth coordinate W, namely

$$\mathbf{P} = H\{\mathbf{P}^\omega\} = H\{(X, Y, Z, W)\} = \begin{cases} \left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W}\right) & if\ W \neq 0 \\ direction(X, Y, Z) & if\ W = 0 \end{cases} \tag{2.5}$$

It is difficult to visualise the mapping from 4-D to 3-D space. For the purpose of illustration, Figure 2.5 shows a mapping of a point from homogeneous form to Euclidean form.

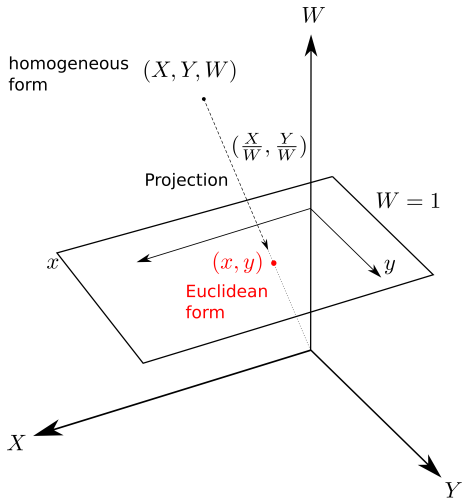


Figure 2.5: The mapping of a point from homogeneous form to Euclidean form.

Based on this idea, a NURBS surface can be expressed in a homogeneous form as

$$\mathbf{S}^{\omega}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}^{\omega}_{i,j} \qquad 0 \leq u,v \leq 1, \tag{2.6}$$

where $\mathbf{P}^{\omega}_{i,j} = (\omega_{i,j}x_{i,j}, \omega_{i,j}y_{i,j}, \omega_{i,j}z_{i,j}, \omega_{i,j})$. To obtain the NURBS surface in 3-D space, simply apply the mapping H to every control point.

More details on NURBS and the properties can be found in [51, 52].

2.3 Literature review of parametrisation methods

There are various parametrisation methods, this section will give a detail discussion on these methods. The main literatures will also be surveyed.

2.3.1 CAD-free parametersation

Parametrisation methods that have no link back to the CAD geometry are normally referred to as CAD-free parametrisation. Some of them are more generic, such as node-based method and FFD, while some are designed for specific applications. However, none of these methods can satisfy all the requirements mentioned in Section 2.1.

2.3.1.1 Generic CAD-free parametrisation method

Node-based parametrisation uses the displacement of every node of the surface grid as design variables (see Fig. 2.1), thus provides the richest design space that can be expressed by the CFD mesh. This method is automatically set up through the surface grid, thus is able to use an existing grid for optimisation. In addition, the grid regeneration process can be avoided during the shape optimisation process. However, there are two major disadvantages of this method. Firstly, additional smoothing to prevent noisy surfaces which are not manufacturable is needed [61]. The reason is that, the high-frequency oscillatory modes resulting from reduced regularity of the gradients, are not visible to the flow solver and thus not well suppressed by the optimiser [48]. This should be addressed by regularisation or smoothing of the gradients or displacements [62, 63]. Secondly, the optimised shape exists as a deformed mesh, which needs manual transformation back to CAD systems [64]. During this process, some fine details may get lost thus impairs the optimal performance. The transformation process is also not trivial and straightforward.

Another kind of CAD-free parametrisation is the FFD method. First formally introduced by Sederberg et al. [65], FFD is a well-established technique used in computer graphics for morphing images and deforming models. During recent years, FFD has also been applied in shape optimisation problem as a parametrisation method [43, 66–68]. The basic idea of FFD is embedding an object of interest inside a flexible control volume, which can be deformed by moving the lattices of the control volume. An example of the FFD approach is given in Fig. 2.6. One advantage of FFD is that it can do volume-based deformation, therefore can be extended to consider structural deformations. The other advantage of FFD is that, the computational grids are also automatically deformed when deforming the whole volume around (or inside) the object. However, the FFD method holds some shortcomings. Firstly, they need auxiliary shape grids so their construction would become very cumbersome when rich design spaces are needed for complex configurations. Secondly, human effort is required to set up the design space, leading to a restricted design space which may not include relevant optimum [43]. Finally, same as the node-based method, the optimised shape exists as a deformed mesh which needs to be transcribed to CAD system.

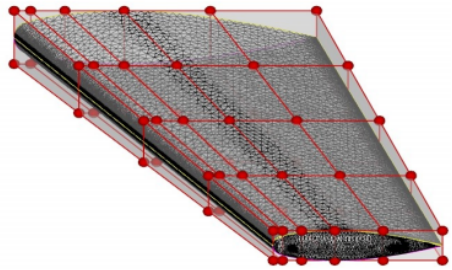


Figure 2.6: A free-form deformation example [69].

The polynomial and spline representations, such as Bézier, B-splines and NURBS, can reduce the total number of design variables, compared to the node-based method. The Bézier form is an effective representation for simple curves, however it does not have the local control property, i.e. moving one control point will deform the whole shape. B-splines form is a better representation, which can be considered as a composite of several Bézier segments. B-splines provide local control property, therefore only a portion of the geometry will be perturbed if a control point is modified. The drawback of B-splines representation is that, they cannot exactly represent conic and circular sections (for instance, the half-cylinder in Fig. 2.1), which are quite common in engineering applications. Thus, approximation is needed with B-splines while this problem is overcome by NURBS. The studies on application of NURBS in shape optimisation problem will be summarised in Section 2.3.3.

2.3.1.2 Airfoil parametrisation method

Some parametrisation methods are designed specifically for particular applications, for example, the airfoil parametrisation. One method of this kind is the Hicks-Henne shape functions [11], which defines the airfoil as the base airfoil plus a linear combination of basis functions, i.e.:

$$y = y^{base} + \sum_{i=1}^{N} a_i f_i, \tag{2.7}$$

where y^{base} is the base airfoil, a_i and f_i are coefficients and basis functions, respectively. The basis function proposed by Hicks and Henne [11] are the sine functions:

$$f_i = sin^{t_i}(\pi x^{ln(0.5)/ln(h_i)}), \tag{2.8}$$

where h_i locates the maximum point of the bump and t_i controls the width of the bump.

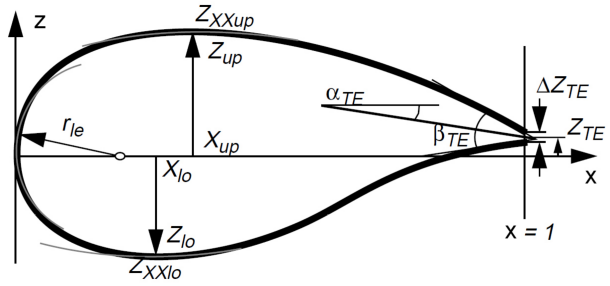


Figure 2.7: A geometric representation of the parameters used in PARSEC method [70].

An alternative approach is the Parameterised Sections (PARSEC) method developed by Sobieczky [70], which expresses the airfoil shape as a liner combination of suitable base function. 12 meaningful properties (shown in Fig. 2.7), such as the upper/lower crest position and max thickness, are then selected to close the system. Specifically, the upper and lower surfaces are defined as:

$$\begin{aligned} y_{upper} &= \sum_{i=1}^{6} a_i x^{i-0.5} \\ y_{lower} &= \sum_{i=1}^{6} b_i x^{i-0.5} \end{aligned} \tag{2.9}$$

where y_{upper} and y_{lower} are y coordinates of points on the upper and lower surface, respectively. a_i and b_i are linear combination coefficients. The advantage of this method is that no baseline shape is needed. In addition, more intuitive control of the shape is allowed since real geometric properties are used. However, the design space is limited to only 12 design variables.

In past several years, a method based on singular value decomposition (SVD) has been proposed as a new parametrisation method for airfoil [71–74]. The basic idea is to apply SVD to a training library consists of many airfoils, say hundreds or thousands, such that some dominant deformation modes are extracted out. Then, these modes compose the design space through linear combination, and the combinational coefficients are used as design variables. In this way, the number of design variables is greatly reduced while still covering a large design space. The other advantage of this method is that, the training library consists of nice airfoils accumulated in past many years, thus design experiences are involved, such that good performance can be obtained. This idea has been applied successfully in several studies [75–78]. However, it seems that this method is more suitable for airfoils thus so far most of these work are limited to 2-D airfoil shape optimisation. Until recently, Allen et al. [79] tried to extend this idea to 3-D shape

optimisation problem. In addition, since a training library consists of many airfoils is needed, this method is non-trial to set up.

Another very similar idea is using Principal Component Analysis (PCA) to reduce the number of design variables [80]. This method also builds the design space based on a large amount of data of shapes accumulated in industrial fields. The method is very helpful in gradient-free optimisation problems, however in gradient-based methods where the adjoint method (see Chapter 3) is chosen to calculate gradients, its strength is limited. In addition, although there was an attempt to apply this method to the optimisation of film cooling hole [80], its main applications are still limited to airfoil optimisation due to the requirement of a trailing library. Another idea sharing the same features is the method based on Proper Orthogonal Decomposition (POD) [81, 82].

The methods presented in this section (Hickes-Henne shape functions, PARSEC, SVD and PCA-based) are suitable for airfoil parametrisation, but it is difficult to generalize them for general geometries, such as the S-bend and U-bend used in later chapters. Therefore, the application of these methods are limited. One can find more details on airfoil parametrisation methods and their comparison in [83–85].

2.3.2 CAD-based parametrisation

In CAD-based methods, the parametrisation of shape is a part of the CAD model which is kept inside the design loop. Hence, the resulting optimal shape is directly available for further analysis and manufacturing. According how the parametrisation is defined, CAD-based parametrisation can be divided into explicit and implicit methods. The main difficulty for gradient-based optimisation with CAD-based parametrisation then is that, it is also needed to compute derivatives of the parametric CAD model.

2.3.2.1 Explicit CAD-based parametrisation

The parametrisation can be defined *explicitly* in a parametric CAD system, as often done to generate a family of parts in different sizes or for manual design space exploration. A number of researches have been performed utilising explicit CAD-based parametrisation methods in shape optimisation, and it can be found that there are several ways to implement explicit CAD-based optimisation, i.e.:

- Firstly, one can develop application-specific parametrisation tools, like in the work of Gräsel et al. [86] or Verstraete [87] as well as Torreguitart et al. [88]. These tools take extensive engineering experience into the account, thus allowing to use

conventional parameters, such as trailing/leading edge radius and wing space, to parametrise geometries and act as design variables.

- Secondly, commercial CAD software can be utilised, such as CATIA V5 [89] and Siemens NX[1]. For example, Fudge et al. [90] presented a CAD-based geometry control system through the Computational Analysis PRogramming Interface (CAPRI), which is an interface to different CAD systems. By using CAPRI, one can use CAD parameters as design variable. Similar CAD-based methods through CAPRI have also been reported by other researchers [91, 92]. Nemec et al. [93] directly employ the parametrisation defined in the Pro/ENGINEER Wildfire CAD system[2]. Robinson et al. [94–96] utilised the CAD parameters defined in CATIA V5, such as sketch-based and dress-up features, as design variables. Vasilopoulos et al. [97] also used a closed-source commercial CAD tool to optimise a compressor stator at multiple operating conditions.

- In addition, open-source CAD kernel, e.g. Open Cascade Technology (OCCT)[3] [98]. Dannenhoffer from New York and Haimes from MIT followed this way and used OpenCASCADE to define their CAD-based parametrisation [99]. Also, Auriemma et al. [45], Banovi et al. [100] and Mykhaskiv et al. [101] tried to integrate OpenCASCADE into optimisation loop as CAD-based parametrisation.

These methods use CAD representation of the geometry at each stage of the cycle so that the CAD system can be integrated into the design loop. As mentioned before, it is required to compute CAD sensitivities in gradient-based optimisation with CAD-based parametrisation. The derivative computation of the explicitly parametrised CAD model can then be achieved through finite differences (FD), i.e. for commercial CAD software [94, 95, 102, 103] or OpenCASCADE [99], which incurs the typical problems of FD with errors, choice of step size. Vasilopoulos et al. [103] and Agarwal et al. [96] avoid problems of robustness due to patch renumbering by approximating the surfaces with triangulations (STL), which in turn has issues with projection near sharp corners. Another option of computing gradients is applying automatic differentiation (AD) to the source code of the open-source CAD system, for instance the OpenCASCADE, or in-house CAD tool, which addresses issues of accuracy and will allow use of the efficient reverse mode [45, 88, 100, 101]. This approach still requires a lot of engineering experience to define a suitable CAD parametrisation.

[1] https://www.plm.automation.siemens.com/zh/products/nx/
[2] http://www.ptc.com/community/landing/wf3.htm
[3] https://www.opencascade.com/

While explicit CAD-based parametrisation has the advantage that geometric constraints such as thicknesses or radii can be directly built into the design space, there are several drawbacks of these methods. Firstly, these parametrisations typically either do not have sufficient freedom to capture relevant modes, or require important human knowledge about the flow and incur significant user effort to set up, which ultimately negates the benefits of numerical optimisation. Secondly, these parametrisations are normally defined using certain CAD system which in general is not transferable among systems. In addition, for some geometries, it is not an easy task to use explicit parameters, such as thickness and width, to represent the shape while maintaining rich design space. An example is the S-bend air duct from automotive industry shown in Fig. 7.1, whose design surfaces are the middle cranked section. Finally, and more severely, current CAD systems in general are proprietary, which do not offer derivatives of surface displacements with respect to the design variables needed in the chain rule.

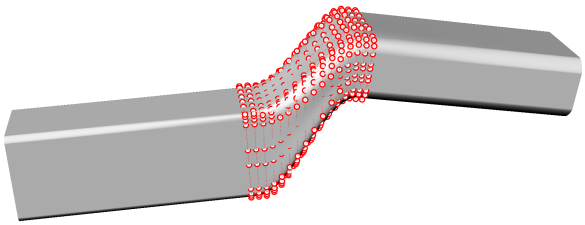


Figure 2.8: An S-bend air duct from automotive industry.

2.3.2.2 Implicit CAD-based parametrisation

In addition to explicit CAD-based parametrisation, a CAD-based parametrisation can also be defined *implicitly*, i.e. to arise from the CAD model's generic description such as the collection of NURBS patches in the STEP (see Section 5.8) or IGES standards [104]. The positions and weights of the control points (CP) of the NURBS patches can be used as design variables.

There are also many works in the literature following this paradigm. In [69], Andres et al. gave a comparison between two kinds of CAD-based parametrisation approaches, one of which is using NURBS to describe the geometry. In this work, it was claimed that the geometry information were extracted from IGES file created by CATIA V5. However, only a 2-D airfoil optimisation was performed and how the CAD system was integrated into the design loop was not described. An improved approach was proposed by Martin et al. in [50, 105], where geometry was represented using NURBS, and the IGES format was used to interchange information between different systems, so that fast incorporation

of the optimisation procedure into existing design chains was possible. The continuous adjoint methodology was used for fast gradient computation. However, this approach can only be applied to geometries with one single moveable patch or only have deformation inside a patch, the continuity between adjacent patches can not be maintained. Mueller et al. [106] integrated the CAD parametrisation into the design chain by extracting information from the STEP file, and the method is applied to a multipoint optimisation of a turbocharge radial turbine.

This approach offers a number of advantages. Firstly, only a subset of CAD functionality is needed which can straightforwardly be implemented in light-weight standalone tools, which in turn significantly simplifies the derivative computation with AD tools [48]. Secondly, the implicit parametrisation through control points typically produces a suitably rich design space without the need for manual setup [107]. Thirdly, provided that the design space remains coarser than the CFD mesh, gradient regularisation as needed for mesh-based parametrisations [62] is not required. On the downside, a methodology needs to be introduced for imposition of geometric constraints, such as continuity constraints between NURBS patches or thickness, box and radii constraints.

Note that since STEP file is based on NURBS, thus more studies of implicitly defined CAD-based methods will be surveyed in Section 2.3.3.3. The CFD optimisation group at QMUL has also performed researches on implicitly defined CAD-based methods. These works will also are summarised in Section 2.3.3.

2.3.3 Application of NURBS in shape optimisation

This section summarises past studies on the application of NURBS in shape optimisation, including structural optimisation and CFD-based shape optimisation. NURBS could be applied in two aspects. First, use NURBS to represent a geometry. Second, add weights into design space during optimisation.

NURBS have been widely used in shape representation and optimisation, since there are several advantages of NURBS over other parameterization techniques. Firstly, NURBS can integrate with standard CAD systems seamlessly, because NURBS are the standard of the STEP file to describe geometry. Secondly, NURBS can support very wide range of geometries, namely they can not only support free-form surface but also implicit surface. Compared to B-splines, which is also widely used, NURBS can describe conic and circular shape accurately. Therefore, most geometries can be handled by using a unified framework with NURBS. More importantly, the weights of NURBS control points provide additional freedom to adjust the shape, thus it is possible to obtain better and

smoother shape in design optimisation.

One thing should note is that, although NURBS is the standard of most CAD systems, not all the parametrisation methods using NURBS are CAD-based, as shown in Fig. 2.2. For example, those methods which only use NURBS curve to describe airfoils but do not involve any CAD systems in the optimisation loop [108–110], should be better regarded as the spline-based approach [53].

2.3.3.1 Application of NURBS in structural shape optimisation

NURBS have been extensively applied in structural optimisation, especially in isogeometric analysis (IGA) [111, 112]. This is because NURBS can play a bridge between CAD model and the Finite Element Method (FEM) model, so saving man power from manually adapting FEM meshes to CAD models [113]. For example, Ziani et al. [114] discussed the effect of weights on structural shape optimisation and claimed that by using weights better optimisation results can be obtained. However, in this paper, very few details were presented, therefore it almost gave no clue on how to use weights and how the results are obtained. Song et al. [115] investigated how both the positions and weights are used as design variables in IGA. However, in this work, positions and weights were used alternatively rather than simultaneously. In [28–30], the homogeneous form of NURBS was applied to the structural optimisation problem to optimise axisymmetric solids and shell structures. The homogeneous form of NURBS allows both position and weights of control points to be used as design variables. These works have shown that NURBS work well together with the FEM.

There are also some works performed utilising NURBS in CFD-based shape optimisation problem. A part of them belong to CAD-based category, while others are not.

2.3.3.2 Application of NURBS in CAD-free CFD shape optimisation

Regarding studies using NURBS but are CAD-free, one of the main problems is that the design loop is not complete and automatic, thus additional work is needed to exchange data and utilise the optimal results.

Some researchers tried to utilise NURBS to represent geometries more effectively. For example, Mengistu et al. [116] tried to find an optimum representation of generic airfoil geometry using NURBS, which can be used to parametrise wings as well as gas turbine blades. The Simulated Annealing optimisation method was utilised. This work showed initial results in representing airfoils with NURBS. Wessels et al. [117] presented two

methods based on NURBS to represent airfoils, i.e. the top and bottom NURBS scheme and the thickness-camber NURBS scheme. It was shown that both schemes can replicate the given set of airfoils well. Bentamy et al. [59] introduced how to represent a wing with NURBS, but no shape optimisation was performed. Lepine et al. [108, 109] demonstrated that a large variety of airfoils can be expressed within specified tolerance by using 13 NURBS control points. Ma et al. [118] presented an FFD method based on NURBS for complex aerodynamic shape deformation. But in this work, only the local modification property of NURBS in shape deformation was shown, no optimisation was performed.

Studies have also been performed to investigate the performance of NURBS in shape optimisation. In the work of Lepine et al. [108, 109], NURBS were firstly used to approximate airfoils, which were then used in shape optimisation. It was shown that NURBS can produce smoother optimal shape, compared to the shape function parametrisation method. However, in those work, it seems that there was no CAD system involved. In addition, the sensitivity of cost function w.r.t. design variables were calculated using FD, thus the computational cost would become prohibitive quickly as the number of design variables increases. Bentamy et al. [1] presented the usage of NURBS in parametrising a complex 3-D wing surface and its use in the first stage of aerodynamic design, considering the manufacturing tolerances. However, details about the design variables were not given. In addition, the elements of the Jacobian (derivatives) were computed using FD. In [119], Lozano et al. computed the flow sensitivities based on domain and boundary integrals using ONERA M6 wing described with NURBS. However, in that work, no shape optimisation was performed. Another work using NURBS by Painchaud-Ouellet et al. [110] performed optimisation using both single-point and multi-point considering thickness constraint. In addition, both position and weight of the control points are perturbed in that work. However, again FD is used to compute the gradient. What's more, the work was only about optimisation of 2-D airfoil and no CAD-systems were involved. Liang et al. [120] applied NURBS in multi-objective airfoil optimisation problem. Both position and weights of control points were chosen as design variables in that work. However, gradient-free optimiser was used thus the optimisation was restricted to 2-D airfoil case because of the computational cost issue. In a follow-on work of these authors [121], the 3-D ONERA M6 wing was optimised in multi-objective manner via CFD approximation model and genetic algorithm (GA). In that work, only a very small number of design variables (22 design variables) were used in order to reduce computational cost. Therefore, the design space was rather limited. Yildirim et al. [122] performed wing shape optimisation by using NURBS and discrete adjoint method. Although they claimed that NURBS control points were used, no evidence were given in the paper. In addition, in that work, they did not give information about where NURBS data came from, and whether any CAD systems were involved in the optimisation loop.

As can be been, current researches on applying NURBS in CAD-free shape optimisation hold some shortcomings, which are listed as following:

- The parametrisation has no direct connection to the CAD model. Reverse engineering techniques would necessary to reconstruct the CAD model from the optimised grid [64, 123], which may impair the optimality of design.
- FD is utilised to compute gradients, or gradeint-free optimisation methods are employed. In both ways, the number of design variables would be limited due to the issue of computational costs.

2.3.3.3 Application of NURBS in CAD-based CFD shape optimisation

With regard to application of CAD-based parametrisation method via NURBS in CFD shape optimisation, one can also find some studies in the literature. Andres et al. [69] presented a preliminary comparison between FFD-based and NURBS-based parametrisation method. However, they actually did not demonstrate the CAD-based property of their methods. In addition, only the position of control point was used, weight adjustment was not considered. Martin et al. [50, 105] reported an automated optimisation framework consists of DLR TAU code and CAD-based parametrisation method with NURBS. They claimed that their optimisation framework was based on NURBS, but in these papers, they stated clearly that weights were not touched. Therefore, B-Splines were actually used in their work. In addition, although the continuous adjoint method was used in these work to save the computational cost of computing gradients, the derivation of continuous adjoint solver was tedious (see Section 4.3.1). In a follow-on work of Martin et al. [124], a control box approach based on NURBS to calculate gradients was presented. They claimed that it was easy to handle continuity by using the NURBS control box approach, but no details and results about the continuity were presented. In a recent work, Verstraete et.al [46] employed tri-variate B-splines to parametrise the U-bend cooling passage based on CAD format. The authors revealed that by using the tri-variate B-splines the mesh deformation process was much easier. However, it seems that it is not trivial to create and set up this parametrisation for a complex geometry.

In addition to above researches, Yu et al. [47] proposed to extract NURBS information from the STEP file, and demonstrated the effectiveness using a 2-D airfoil case. The method was extended to 3-D case to handle the geometric continuity constraints between different patches [48, 49], and to compute intersection lines [125]. In [101], Mykhaskiv et al. compared NURBS-based and parametric-based parametrisations using the differentiated OpenCASCADE, and suggested that NURBS-based method was more suitable

for optimising non-conventional components. However, although termed with NURBS, in these works actually only B-splines were supported, thus the benefits of NURBS were not utilised. For instance, many STEP files exported by CAD software were not supported, and weights of control points were not varied during optimisation. In addition, as will be shown in Section 5.4.7, in these works the number of design variables may change during optimisation, hence advanced optimisers such as BFGS cannot be applied directly. Thus, measures must be taken to solve this. Besides, although the human efforts needed were reduced significantly, the good performance of these approaches still heavily reply on user's experiences.

2.3.3.4 Summary of application of NURBS in shape optimisation

Based on above discussions, one can see that most works of using NURBS in shape optimisation focus on CAD-free parametrisation What's more, while some studies have been conducted on the application of NURBS in shape optimisation problem, there is still a lack of method and an optimisation framework holding the following features:

- CAD-based, namely both input and output files are compatible with CAD software and can be exchanged among different systems.
- Can be applied to 3-D shape optimisation problem.
- Can handle geometric constraints, such as geometric continuity between patches.
- Utilise the benefits of NURBS, namely support wider range of geometries, and include both position and weight of control points in the design space.
- Use gradient-based optimisation approaches, and support different optimisers.
- Adjoint-based, such that computational inexpensive gradients are available.
- Automated, thus do not require additional user efforts during the optimisation.
- Easy to set up, do not need many efforts to set up, and do not reply on user's expertise heavily to obtain good performance.

This thesis will focus on CAD-based parametrisation method by using NURBS, and will develop method and optimisation framework to fulfil above requirements.

2.4 Summary

This chapter introduces geometry parametrisation, which is a key aspect in shape optimisation, in detail. Firstly, requirements for a good parametrisation method are given. Secondly, the NURBS surface and their homogeneous form are discussed. Then, an extensive literature review of various parametrisation approaches is given with emphasis on two major families of parametrisation methods, namely CAD-free and CAD-based approaches. In addition, the applications of NURRBS in shape optimisation are also surveyed.

Compared to their CAD-free counterparts, CAD-based parametrisation methods can return a CAD format, which is desired by industry, thus are normally preferred. There are different ways to implement CAD-based parametrisation, i.e. explicit parametrisation based on parametric CAD systems and implicit parametrisation based on CAD model's generic description like STEP file. The main features of implicit parametrisation are that they do not heavily rely on engineering experience, and large number of design variables are possible. In addition, it is easier to obtain accurate and robust derivatives. By utilising NURBS, more benefits can be achieved. Therefore, the explicit CAD-based parametrisation is more suitable for cases where decent parametrisation is available from engineering experience, while the implicit methods are preferred when optimising non-conventional shapes and exploring new designs.

Based on infomation presented in this chapter, this thesis will focus on CAD-based parametisation methods. To be more specific, this work will mainly investigate CAD-based method via NURBS repersentation.

Chapter 3

Numerical optimisation

CFD-based shape optimisation is an important application of numerical optimisation technique, which has attracted widespread interest in recent years. This chapter firstly gives a general introduction to numerical optimisation in Section 3.1, then introduces the shape optimisation problem based on CFD in Section 3.2. Some basis aspects about numerical optimisation, such as optimiser and line search algorithm, are discussed in Section 3.3. Section 3.4 is then devoted to summarise gradient computation methods, with emphasis on AD.

3.1 Introduction

Numerical optimisation algorithms are increasingly used in advanced design problems. Based on whether gradient information is needed in the optimisation or not, there are two main kinds of optimisation methods: gradient-free (stochastic) and gradient-based (deterministic) optimisation method. In gradient-free optimisation only the cost function value is needed, while in gradient-based methods the first derivatives even the second derivatives are also required.

Each approach has its advantages and drawbacks, and the choice is problem dependent. Gradient-free optimisation methods such as Genetic Algorithms (GA) [126] or Evolutionary Algorithms (EA) [127] are well established, especially for cases where the computational cost of a function evaluation is low, such as the linear structural optimisation. The gradient-free methods are often conveniently used as black-box packages, since no modifications to the simulation tool are needed. However, these methods require numerous function evaluations when going beyond 50-100 design variables [46], and this

cost becomes prohibitive when used with expensive CFD models. In many cases a single evaluation of the CFD problem may require several hours or days of run time. Therefore, gradient-free approaches are not suitable in shape optimisation with CFD if there are many design variables.

Since the duration of the complete CFD-based optimisation process is "(number of CFD evaluations) × (duration of a single CFD evaluation)" [18], we hence have to reduce the number of evaluations as much as possible to save computational time. Therefore, in shape and topology optimisation with CFD, gradient-based methods have been adopted as the method of choice because their convergence suffers much less from large design spaces. They converge to the optimum in many fewer design iterations compared to stochastic approaches. For instance, gradient-based optimisation can lead to an optimum with $O(10)$ CFD evaluations [31, 61].

However, there are also limitations for gradient-based methods. Firstly, the linearisation used in gradient-based optimisation is only valid when the objective function is at least once continuously differentiable. For those cases where design variables are integer values, e.g. the number of engines, stochastic approaches may be better. Besides, the gradient-based methods generally will converge to the nearest local minimum rather than global optimum. A combination of stochastic methods and gradient-based method can be used for global optimisation. Some recent work on multimodality and global optimisation in aerodynamic design can be found in [128–130].

In practical optimisation design, the designer normally start based on an existing design, which is already good. The aim is to obtain some improvements. For example, improve the wing shape based on current aircraft. In this case, the local optimum is often acceptable thus the gradient-based method is suitable. Based on this fact, we focus on using gradient-based approaches in this work.

3.2 Shape optimisation based on CFD

In aerodynamic shape optimisation problems, three components are used to describe the problem: cost function J, state variables $\mathbf{U}$ and design variables $\boldsymbol{\alpha}$. The cost function J is determined by both state variables $\mathbf{U}$ and design variables $\boldsymbol{\alpha}$. The flow variable $\mathbf{U}$ is computed on the spatial discretisation $\mathbf{X}_v$, which deforms smoothly given deformation of the surface mesh discretisation $\mathbf{X}_s$. $\mathbf{X}_s$ is derived from surface $\mathbf{S}$ which is determined by the design variables $\boldsymbol{\alpha}$. Thus, the dependency between cost function J and design

variables $\boldsymbol{\alpha}$ is:

$$J = J(\mathbf{U}(\mathbf{X}_v(\mathbf{X}_s(\mathbf{S}(\boldsymbol{\alpha})))), \boldsymbol{\alpha}). \tag{3.1}$$

The main terms related to numerical optimisation in CFD-based optimisation are introduced below:

cost/objective function: denoted here as J. The cost function is a scalar function that is to be minimised, and it is the measure of optimality. Typical cost functions in aerodynamic optimisation are lift, drag and pressure loss, etc.

flow variable: denoted here as $\mathbf{U}$. Flow variables satisfy the flow equations (see Chapter 4), which are constraints to the optimisation problem.

design variable: denoted here as $\boldsymbol{\alpha}$ for the vector of design variables, while α_i refers to one particular design variable. Design variables are what will be modified during the optimisation, however they do not necessarily control the shape directly. Figure 2.1 shows examples of design variables, which are NURBS control points and surface mesh coordinates, respectively.

gradient/sensitivity of the objective: denoted here as $\frac{dJ}{d\boldsymbol{\alpha}}$. In gradient-based methods, gradients point out a direction of searching optimal solution.

CAD sensitivity: denoted here as $\frac{\partial \mathbf{X}_s}{\partial \boldsymbol{\alpha}}$, is the sensitivity of the surface mesh points w.r.t. the CAD design variables.

optimiser: Optimiser is used to select a set of new values for design variables in order to find the optimum.

The sensitivity of the cost function J with respect to the design variables $\boldsymbol{\alpha}$, can be obtained using chain rule:

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \frac{\partial J}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \boldsymbol{\alpha}}. \tag{3.2}$$

Quite often $\frac{\partial J}{\partial \boldsymbol{\alpha}} = 0$ because the cost function does not explicitly depend on the design variables. For example, in the U-bend cooling channel case (see Chapter 8), the outlet is located very far from the zone affected by the design variables, thus this term is actually 0.

In the case of using a node-based parametrisation as presented in see Section 2.3.1.1, equation (3.2) becomes

$$\frac{dJ}{d\mathbf{X}_s} = \frac{\partial J}{\partial \mathbf{X}_s} + \frac{\partial J}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{X}_s}. \tag{3.3}$$

where the derivatives of cost function can be provided by the adjoint solver (see Chapter 4).

In industrial design, CAD systems are used to build the surfaces of objects and are needed in manufacturing. It is of interest to preserve the CAD format of the surfaces throughout the design process, although sometimes we need to modify the CAD description to make it suitable for optimisation. The parameters of the surface parametrisation, such as the control points of NURBS surface, can then be used as design variables, and their optimal configuration is the target of optimisation. In the context of CAD-based parametrisation, the derivative of the objective function is extended to:

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \frac{\partial J}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{X}_s} \frac{\partial \mathbf{X}_s}{\partial \mathbf{S}} \frac{\partial \mathbf{S}}{\partial \boldsymbol{\alpha}}, \tag{3.4}$$

where the CAD sensitivity $\frac{\partial \mathbf{X}_s}{\partial \boldsymbol{\alpha}}$ in equation (3.4) has to be obtained from the information provided by CAD systems. More details on the computation of CAD sensitivity will be introduced in Section 5.4.6.

Once the gradient is obtained, gradient-based methods can be utilised to perform the optimisation.

3.3 Optimiser and line search

Optimiser is the algorithm used to find a new set of values for the design variables $\boldsymbol{\alpha}$ to search the optimum. Many gradient-based optimisation methods move along a search direction for some distance to find new value of $\boldsymbol{\alpha}$, where the cost function value is lower (for minimisation problem). Specifically, to find

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + s_k \mathbf{p}_k, \tag{3.5}$$

which leads $J(\boldsymbol{\alpha}_{k+1})$ smaller than $J(\boldsymbol{\alpha}_k)$, where the subscript $k+1$ and k are design iterations, $\mathbf{p}_k$ is the search direction, s_k is step length. The important thing here is to firstly find a search direction which can lower the cost function, and then decide how long should move along this direction.

It has been demonstrated that if $\mathbf{p}_k$ is a descent direction, it should satisfy the condition

$$\mathbf{p}_k^T \mathbf{g}_k < 0, \tag{3.6}$$

where $\mathbf{g}_k$ is the gradient of cost function at point $\boldsymbol{\alpha}_k$. Optimisation methods differ mainly

in the way of finding $\mathbf{p}_k$ in each optimisation iteration. For instance, in the steepest descent method, $\mathbf{p}_k$ is the negative gradient direction ($\mathbf{p}_k = -\mathbf{g}_k$). In the Newton method, $\mathbf{p}_k$ is the product of inverse Hessian matrix (matrix of second derivatives) and negative gradient direction ($\mathbf{p}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$). In quasi-Newton method, an approximation to the Hessian matrix with a positive definite matrix is used instead of the exact Hessian matrix.

In following sections, more details will be given on the search direction and step length.

3.3.1 The steepest descent method

The steepest descent method utilises the negative gradient direction as the search direction, i.e.

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k - s_k \mathbf{g}_k. \tag{3.7}$$

It is the simplest gradient-based optimisation methods and can be very easily implemented. It requires only the cost function value as well as the first derivative of cost function.

A typical process of the steepest descent algorithm with inexact line search (see Section 3.3.3) is presented in Algorithm 3.1. As can be seen that, the history information of gradient is not required during the optimisation process. Because of this property and its simplicity, the steepest descent method has been used frequently together with the NSPCC approach previously by Xu et al. [48, 49] and also in this study.

However, the steepest descent method is a first order approach, which may not be very efficient and converges slowly for complex problem. Therefore, more efficient methods have also been explored.

Algorithm 3.1: Steepest descent method with inexact line search

1 $k = 0, 1, 2, \ldots$;
2 $\boldsymbol{\alpha}_0 = \boldsymbol{\alpha}_{initial}$;
3 **repeat**
4 Compute J_k;
5 Set $\mathbf{p}_k = -\nabla J(\boldsymbol{\alpha}_k)$;
6 Calculate s_k^* so that the Wolfe condition is satisfied ;
7 Set $\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + s_k^* \mathbf{p}_k$;
8 **until** $\| \nabla J(\boldsymbol{\alpha}_{k+1}) \|$ *is sufficiently small*;

3.3.2 Quasi-Newton method

If the search direction in equation (3.5) is the product of inverse Hessian matrix with negative gradient, i.e.

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k - s_k \mathbf{H}_k^{-1} \mathbf{g}_k, \tag{3.8}$$

then it gives the Newton method. The Newton method has second order convergence rate such that it can quickly converge to local optimum if it starts near it. However, it is computational expensive to compute the exact Hessian matrix and invert it in each optimisation iteration, leading to the development of another kind of methods, called quasi-Newton methods.

One of the most famous quasi-Newton approaches is the BFGS method, named after its inventors Broyden, Fletcher, Goldfarb, Shanno [131]. The main idea of BFGS algorithm is to approximate the Hessian matrix (or its inverse) with a positive definite matrix, avoiding the expensive computations of the Hessian matrix. The approximation matrix is updated in each iteration with a low-rank update. The BFGS approximation of the inverse Hessian ($\mathbf{B} \approx \mathbf{H}^{-1}$) is obtained by [131]:

$$\mathbf{B}_{k+1} = (\mathbf{I} - \boldsymbol{\rho}_k \boldsymbol{\delta}_k \mathbf{y}_k^T)\mathbf{B}_k(\mathbf{I} - \boldsymbol{\rho}_k \mathbf{y}_k \boldsymbol{\delta}_k^T) + \boldsymbol{\rho}_k \boldsymbol{\delta}_k \boldsymbol{\delta}_k^T, \tag{3.9}$$

where $\boldsymbol{\delta}_k = \boldsymbol{\alpha}_{k+1} - \boldsymbol{\alpha}_k$, $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$, $\boldsymbol{\rho}_k = \dfrac{1}{\mathbf{y}_k^T \mathbf{s}_k}$. Then we have

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k - s_k \mathbf{B}_k \mathbf{g}_k. \tag{3.10}$$

The BFGS approach is more efficient compared to the steepest descent method, while it loses the quadratic convergence of the Newton method [132]. A line search is needed in the BFGS algorithm to safeguard the optimisation process, as shown in Algorithm 3.2:

Algorithm 3.2: BFGS method

1 Choose $\boldsymbol{\alpha}_0$ as an initial estimate of the minimum of $J(\boldsymbol{\alpha})$;
2 Choose $\mathbf{H}_0$ as an arbitrary symmetric positive definite matrix, normally $\mathbf{H}_0 = \mathbf{I}$;
3 $k = 0, 1, 2, \ldots$;
4 **repeat**
5 Set $\mathbf{g}_k = -\nabla J(\boldsymbol{\alpha}_k)$;
6 Set $\mathbf{p}_k = \mathbf{H}_k \mathbf{g}_k$;
7 Find s_k, so $J(\boldsymbol{\alpha}_k + s_k \mathbf{p}_k)$ satisfies Wolfe conditions ;
8 Set $\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + s_k \mathbf{p}_k$, $\boldsymbol{\gamma}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$ $\boldsymbol{\delta}_k = \boldsymbol{\alpha}_{k+1} - \boldsymbol{\alpha}_k$;
9 Obtain a new positive definite matrix $\mathbf{H}_{k+1}$ such that $\mathbf{H}_{k+1}\boldsymbol{\gamma}_k = \boldsymbol{\delta}_k$
10 **until** $\| \nabla J(\boldsymbol{\alpha}_{k+1}) \|$ *is sufficiently small*;

There are some variants of the BFGS method, such as L-BFGS [133, 134] and L-BFGS-B [135–137]. The problem of using these L-BFGS-based packages in aerodynamic shape optimisation is that, the initial perturbation step is too large for the mesh movement algorithm thus the mesh will be broken. Measures must be taken to deal with this, which is however not trivial because these packages are normally utilised like a black-box tool. Interested readers can refer to those work.

3.3.3 Line search

As mentioned earlier, one should find a step length to decide the distance move along the search direction. This process is called *line search*. To be more specific, line search is the process of finding s_k, so that $J(\boldsymbol{\alpha}_k + s_k \mathbf{p}_k)$ is smaller than $J(\boldsymbol{\alpha}_k)$ to a satisfactory extent.

There are two kinds of line search, the first one is called perfect or exact line search, which chooses s_k^* to minimise $J(\boldsymbol{\alpha}_k + s_k \mathbf{p}_k)$. The other one is weak or inexact line search, which accepts any s_k that make $J(\boldsymbol{\alpha}_k + s_k \mathbf{p}_k) - J(\boldsymbol{\alpha}_k)$ negative and bounded away from 0. Since perfect line search is computationally expensive, therefore in practice and also in this work, the inexact line search is preferred and used.

The Wolfe theorem [138] provides precise conditions on s_k that guarantee convergence of optimisation problems. There are basically three Wolfe conditions which are given as following.

The first Wolfe condition is:

$$\mathbf{p}_k^T \mathbf{g}_k \leq -\eta_0 \parallel \mathbf{p}_k \parallel \parallel \mathbf{g}_k \parallel, \tag{3.11}$$

where η_0 is a small positive constant with typical value 0.01. This condition is stronger than equation (3.6) and requires the angle between $\mathbf{p}_k$ and $-\mathbf{g}_k$ in the range of $(-\frac{\pi}{2} + acos(\eta_0), \frac{\pi}{2} - acos(\eta_0))$.

The second Wolfe condition is:

$$J(\boldsymbol{\alpha}_k + s_k \mathbf{p}_k) - J(\boldsymbol{\alpha}_k) \leq \eta_1 s_k \mathbf{p}_k^T \mathbf{g}_k \tag{3.12}$$

where η_1 is a constant between 0 and 0.5. Typically, $\eta_1 = 0.1$. This condition is used to make sure that s_k can produce a meaningful reduction in the cost function. Also, this condition requires that the step size is not too large.

The third Wolfe condition is:

$$|J(\boldsymbol{\alpha}_k + s_k\mathbf{p}_k) - J(\boldsymbol{\alpha}_k) - s_k\mathbf{p}_k^T\mathbf{g}_k| \geq \eta_2|s_k\mathbf{p}^T\mathbf{g}_k|, \tag{3.13}$$

where η_2 is a constant between 0 and 0.5. This condition ensures s_k is bounded away from 0.

Defining the ratio

$$D(s_k) = \frac{J(\boldsymbol{\alpha}_k + s_k\mathbf{p}_k) - J(\boldsymbol{\alpha}_k)}{s_k\mathbf{p}_k^T\mathbf{g}_k}, \tag{3.14}$$

then the Armijo line search algorithm [132] based on the second and third Wolfe condition is given as in Algorithm 3.3. The Armijo line search method is used throughout this study.

Algorithm 3.3: The Armijo line search algorithm

Choose constants $C > 1, c < 1$, and η_1, η_2 so that $0 < \eta_1, \eta_2 < 0.5$. Typically, $\eta_1 = \eta_2 = 0.1$;
Set $s_k = 1$ and $s_{min} = 0$;
Compute $J(\boldsymbol{\alpha}_k)$, $\mathbf{g}_k$;
Set $\mathbf{p}_k = -\mathbf{g}_k$;
Compute $J(\boldsymbol{\alpha}_k + s_k\mathbf{p}_k)$, $D(s_k)$;
while $(|1 - D(s_k)| < \eta_2)$ **do**
    Set $s_{min} = s_k, s_k = Cs_k$;
    Compute $J(\boldsymbol{\alpha}_k + s_k\mathbf{p}_k)$, $D(s_k)$;
end
while $(D(s_k) < \eta_1)$ **do**
    Set $s_k = s_{min} + c(s_k - s_{min})$;
    Compute $J(\boldsymbol{\alpha}_k + s_k\mathbf{p}_k)$, $D(s_k)$;
end

3.4 Gradient calculation

The key requirement of using gradient-based optimisation is the computation of gradients. Typically, there are several methods that can be used to calculate the gradients: finite differences (FD), complex variables, tangent linearisation, automatic differentiation (AD) and the adjoint method. The first four types will be introduced in this section and the adjoint method will be discussed in Section 4.3.1, because the adjoint method is more related to flow solvers in this study.

3.4.1 Finite differences

The simplest method to compute gradient is finite differences (FD). The derivative of cost function J with respect to a particular design variable α_i is:

$$\frac{\partial J}{\partial \alpha_i} = \frac{J(\boldsymbol{\alpha} + h\mathbf{e}_i) - J(\boldsymbol{\alpha})}{h} + O(h), \tag{3.15}$$

where $\mathbf{e}_i$ denotes the ith unit vector, h is the perturbation step size. The truncation error is $O(h)$, hence this is a first-order approximation.

The derivative with second-order accuracy can be obtained by central finite difference:

$$\frac{\partial J}{\partial \alpha_i} \approx \frac{J(\boldsymbol{\alpha} + h\mathbf{e}_i) - J(\boldsymbol{\alpha} - h\mathbf{e}_i)}{2h} + O(h^2). \tag{3.16}$$

The main advantage of FD is the unparalleled simplicity, especially it can be used straightforwardly with black-box commercial solvers and CAD systems, for example in some CAD-based parametrisations [94, 103]. However, FD suffers from two important shortcomings. Firstly, it is hard to choose a proper step size h. As illustrated in Fig. 3.1, if h is too large the truncation error will corrupt the result, while if h is too small the round-off error will dominate. Another problem with the step size is that, the scale of elements in the design variable vector may be different, thus the proper step size for each variable may not be the same. This makes the selection of the proper step size more difficult and time consuming. Secondly, if FD is used to compute flow sensitivities in CFD shape optimisation, each design variable will incur an additional solve of the flow field (or two in central finite difference) at every design iteration, thus the cost will quickly become prohibitive when the number of design variables increases. This kind of method is called 'state-gradient' method [139]. Researchers from Queen's University of Belfast (QUB) [94, 95, 103] computed the flow sensitivity with adjoint method (see Section 4.3.1) and applied FD to CAD systems for geometry sensitivities, avoiding the prohibitive costs. They also coped with problems of robustness due to patch renumbering by approximating the surfaces with triangula-tions (STL), which in turn has issues with projection near sharp corners.

Due to its simplicity, the FD gradient is utilised to compare with gradients from other methods in this work.

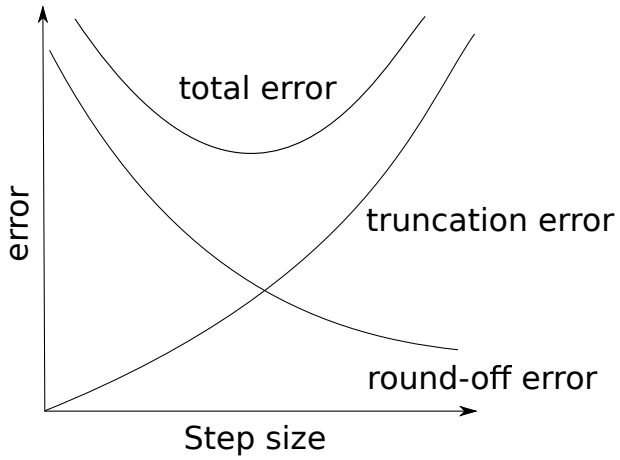


Figure 3.1: Finite differences error dependence on step size.

3.4.2 Complex-step method

One alternative to FD is the complex-step method based on complex variables. By using a pure imaginary step sj, the Taylor expansion of a function f of at point x gives:

$$f(x+sj) = f(x) + s\frac{\partial f}{\partial x}j - \frac{s^2}{2!}\frac{\partial^2 f}{\partial x^2} - \frac{s^3}{3!}\frac{\partial^3 f}{\partial x^3}j + \frac{s^4}{4!}\frac{\partial^4 f}{\partial x^4} + \dots, \tag{3.17}$$

where j is the imaginary unit and s denotes the step size. The derivative of f with respect to x is obtained by equating the imaginary parts:

$$\mathrm{Im}[f(x+sj)] = s\frac{\partial f}{\partial x} - \frac{s^3}{3!}\frac{\partial^3 f}{\partial x^3} + O(s^5), \tag{3.18}$$

where Im means taking the imaginary part. Herewith, we can obtain the gradient:

$$\frac{\partial f}{\partial x} = \frac{\mathrm{Im}[f(x+sj)]}{s} + O(s^2). \tag{3.19}$$

The main advantage of complex-step method is that it avoids the subtraction of equal size parts in FD which gives rise to the round-error when the step size is chosen very small. In addition, as can be seen from equation (3.19), this is a second order estimate of the derivative. The complex-step method provides a tool for comparing the gradients computed using other methods. However, the computational cost still scales with the number of design variables. In addition, there is a significant computational cost due to the complex-variable arithmetic [139].

More details about the complex-step method, such as the implementation and its

connection to other methods, can be found in [140–144].

3.4.3 Tangent linearisation

Consider the Navier-Stokes equations (see Chapter 4) as:

$$\mathbf{R}(\mathbf{U}, \boldsymbol{\alpha}) = 0, \tag{3.20}$$

where $\mathbf{R}$ is the conservative residual of the discretised flow equations, $\mathbf{U}$ are state variables and $\boldsymbol{\alpha}$ is a set of design variables. Taking the derivative of equation (3.20) with respect to $\boldsymbol{\alpha}$, we have:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \boldsymbol{\alpha}} + \frac{\partial \mathbf{R}}{\partial \boldsymbol{\alpha}} = 0, \tag{3.21}$$

which can be written as:

$$\mathbf{A}\mathbf{u} = \mathbf{f}, \tag{3.22}$$

where $\mathbf{A}$ is the Jacobian $\frac{\partial \mathbf{R}}{\partial \mathbf{U}}$, $\mathbf{u}$ is the perturbation field, i.e. the change of the flow field with respect to $\boldsymbol{\alpha}$, $\frac{\partial \mathbf{U}}{\partial \boldsymbol{\alpha}}$, and $\mathbf{f}$ is is the change in residual w.r.t. changes in shape, $\frac{\partial \mathbf{R}}{\partial \boldsymbol{\alpha}}$.

As stated in equation (3.4), the derivative of the cost function J w.r.t. design variable $\boldsymbol{\alpha}$ is:

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \frac{\partial J}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \boldsymbol{\alpha}}. \tag{3.23}$$

Specifically, the derivative of the cost function w.r.t. a particular design variable α_i is:

$$\frac{\partial J}{\partial \alpha_i} = \frac{\partial J}{\partial \alpha_i} + \frac{\partial J}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \alpha_i}. \tag{3.24}$$

which indicates that for each design variable an expensive solve of the perturbation field $\frac{\partial \mathbf{U}}{\partial \alpha_i}$ is required. Therefore, when the number of design variables increases, the computational cost will quickly become too expensive.

3.4.4 Automatic differentiation

Apart from the methods mentioned above, the gradients of a function can also be computed via the approach termed algorithmic differentiation, which is a technique to numerically evaluate derivatives of a function from its source code. This method is based on the fact that every computer program consists of a sequence of elementary arithmetic operations, such as addition and subtraction, and elementary functions, such as exponential and logarithm. Thus, by analytically differentiating these operations and functions, and then assembling the elemental derivatives based on the chain rule, derivatives of

function can be computed. The main attraction of algorithmic differentiation is that it can give very accurate derivatives, namely accurately to machine precision, for functions implemented in programming language.

There are two modes of algorithmic differentiation: forward mode and reverse mode [145]. In forward mode, the function is firstly broken into the most elemental parts. Then, these most elemental parts are differentiated from first to last and assembled based on the chain rule to form the derivatives of the function w.r.t. each independent variable. In contrast, the differentiation in the reverse mode is in reverse sequence after forming the most elemental parts, i.e. from last to first. To be more specific, the last elemental entity is differentiated w.r.t. all the elemental parts.

Taking a function $f(x, y) = ycosx$ as an example. The derivatives of f can be obtained easily as: $\frac{\partial f}{\partial x} = -ysinx$, $\frac{\partial f}{\partial y} = cosx$. The computational graph of this function is given in Fig. 3.2, where each node denotes a primitive function and the edges present the flow of information. Based on this graph, the code list in forward mode and reverse mode are given in Table 3.1 and Table 3.2, respectively. As can be seen, in forward mode information flows from bottom to up, while in reverse mode it is from up to bottom. It can also be observed that in forward mode the last operations give the derivatives, while in reverse mode $\frac{\partial e_4}{\partial e_1}$ and $\frac{\partial e_4}{\partial e_2}$ are the required derivatives. Note that in forward mode the differentiation operations need to be applied twice to obtain gradients for both x and y. In contrast, in reverse mode one single parse is enough to obtain derivatives w.r.t. both independent variables, which is the advantage of the reverse mode.

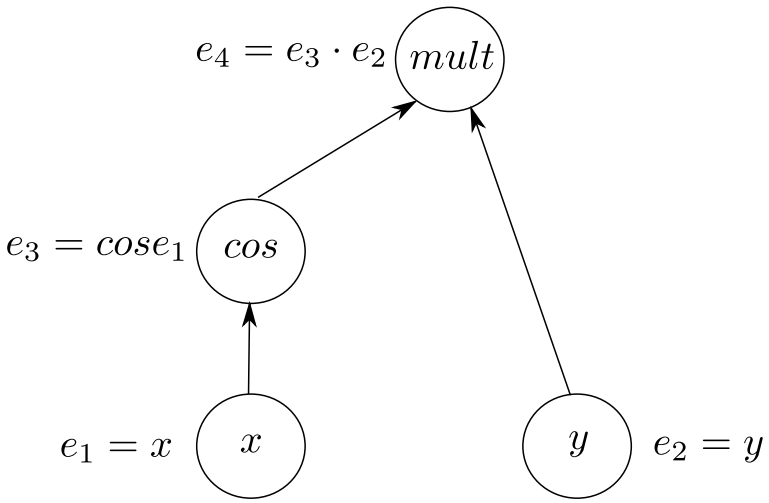


Figure 3.2: The computational graph of function f.

Table 3.1: Forward mode code list of function f.

Forward list	$\nabla = \frac{\partial}{\partial x}$	$\nabla = \frac{\partial}{\partial y}$
∇e_1	1	0
∇e_2	0	1
$\nabla e_3 = -sine_1 \cdot \nabla e_1$	$-sinx$	0
$\nabla e_4 = \nabla e_2 \cdot e_3 + e_2 \cdot \nabla e_3$	$-ysinx$	$cosx$

Table 3.2: Reverse mode code list of function f.

Reverse list	Equivalent evaluation
$\frac{\partial e_4}{\partial e_4} = 1$	1
$\frac{\partial e_4}{\partial e_3} = e_2$	y
$\frac{\partial e_4}{\partial e_2} = e_3$	$cosx$
$\frac{\partial e_4}{\partial e_1} = \frac{\partial e_4}{\partial e_3}\frac{\partial e_3}{\partial e_1} = -e_2 \cdot sine_1$	$-ysinx$

Considering a function which has m dependent variables and n independent variables, $\mathbf{F}(\mathbf{x})$, the forward mode computes

$$\dot{\mathbf{F}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}\dot{\mathbf{x}}, \tag{3.25}$$

which is

$$\begin{bmatrix} \dot{f}_1 \\ \vdots \\ \dot{f}_i \\ \vdots \\ \dot{f}_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_i} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial f_j}{\partial x_1} & \cdots & \frac{\partial f_j}{\partial x_i} & \cdots & \frac{\partial f_j}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_i} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_i \\ \vdots \\ \dot{x}_n \end{bmatrix}. \tag{3.26}$$

It can be seen that, for n independent variables one needs to invoke the differentiated chain n times and a column (shown in red) of the Jacobian matrix is computed each time.

In reverse mode, we compute

$$\bar{\mathbf{x}} = \bar{\mathbf{F}}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}, \tag{3.27}$$

which is

$$\begin{bmatrix} \bar{x}_1 & \cdots & \bar{x}_i & \cdots & \bar{x}_n \end{bmatrix} = \begin{bmatrix} \bar{f}_1 & \cdots & \bar{f}_i & \cdots & \bar{f}_m \end{bmatrix} \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_i} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial f_j}{\partial x_1} & \cdots & \frac{\partial f_j}{\partial x_i} & \cdots & \frac{\partial f_j}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_i} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}. \tag{3.28}$$

It can be seen that in reverse mode one row (shown in green) of the Jacobian matrix is computed with each invocation of the differentiated chain. Therefore, the forward more is much more efficient if the number of output variables is larger than that of input variables. In contrast, the reverse mode is more efficient if the number of input variables is far more than that of output variables.

In the past years, a number of software tools have been developed to produce derivatives codes automatically, making the algorithm differentiation becomes automatic differentiation (AD). With these tools, AD has been applied to numerical optimisation, adjoint codes derivation [41, 42, 146] and other areas such as machine learning [147–149].

There are two different methods to implement AD. One is source code transformation (S-T), the other one is operator overloading (O-O). By using S-T, the source code of a function is replaced by new source code calculating the derivatives. This new source code is generated automatically, and contain the statements calculating derivatives interleaved with original instructions. In the second case of using O-O, the form or sequence of operations in the original source code don't need to be changed, but the operators and simple data types such as double have to be overloaded to obtain the derivatives.In the present work, S-T is used due to the fact that it allows one to view and read the derivative code, and then optimise or modify the code if necessary.

AD will be extensively exploited in this thesis. The AD tool Tapenade[1] [150], which is a tool based on S-T, is utilised to perform AD. Currently it supports Fortran 77, Fortran 90/95 and C code. More details about Tapenade can be found in Appendix D.

Closing this section, the source code of the function $f(x, y) = ycosx$ is used to explain how Tapenade works. $f(x, y)$ is implemented in Fortran 90 as:

```
1 subroutine function (x,y,f)
2    real(8), intent(in) :: x,y
3    real(8), intent(out) :: f
4    f = y*cos(x)
5 end subroutine
```

[1] http://www-sop.inria.fr/tropics/tapenade.html

By stating the input (x,y) and output (f) and calling Tapenade in forward mode, one can obtain the sensitivity code as:

```
1  SUBROUTINE FUNCTION_D(x, xd, y, yd, f, fd)
2     IMPLICIT NONE
3     REAL*8, INTENT(IN) :: x, y
4     REAL*8, INTENT(IN) :: xd, yd
5     REAL*8, INTENT(OUT) :: f
6     REAL*8, INTENT(OUT) :: fd
7     INTRINSIC COS
8     fd = yd*COS(x) - y*xd*SIN(x)
9     f  = y*COS(x)
10 END SUBROUTINE FUNCTION_D
```

The suffix 'D' or 'd' in the subroutine name denotes forward mode. This code needs to be called twice with different argument values to obtain derivatives w.r.t. both x and y. To be more specific, one should set the values of xd and yd, which are called seeds, to obtain the derivatives returned in fd, as shown in the following Listing:

```
1  ! df/dx in forward mode
2  xd=1.0
3  yd=0.0
4  call FUNCTION_D(x, xd, y, yd, f, fd)
5  dfdx=fd
6
7  ! df/dy in forward mode
8  xd=0.0
9  yd=1.0
10 call FUNCTION_D(x, xd, y, yd, f, fd)
11 dfdy=fd
```

By running Tapenade in reverse mode, the sensitivity code obtained is:

```
1  SUBROUTINE FUNCTION_B(x, xb, y, yb, f, fb)
2     IMPLICIT NONE
3     REAL*8, INTENT(IN) :: x, y
4     REAL*8 :: xb, yb
5     REAL*8 :: f
6     REAL*8 :: fb
7     INTRINSIC COS
8     yb = COS(x)*fb
9     xb = -(y*SIN(x)*fb)
10    fb = 0.0_8
11 END SUBROUTINE FUNCTION_B
```

The suffix 'B' or 'b' in the subroutine name means reverse mode. This code then only needs to be invoked once to compute the derivatives w.r.t. both x and y. In this case,

fb is the seed, as shown in the following Listing:

```
1 ! compute df/dx and df/dy in reverse mode
2 fb=1.0
3 call FUNCTION_B(x, xb, y, yb, f, fb)
4 dfdx=xb
5 dfdy=yb
```

Examples of differentiated codes produced by Tapenade on the CAD-based parametrisation method in this study will be presented in Section 6.1.1. To find more details about the principle and application of AD, one can refer to [35, 151–157]. More works on AD at QMUL can be found in [41, 42, 146].

3.5 Summary

Numerical optimisation technique has been introduced in this chapter, since it is a key component of the CFD-based shape optimisation framework. The optimiser together with line search approach will be utilised in practical optimisation cases in this study.

Besides, much attention of this chapter has been paid to introduce methods of computing gradients, because the gradient information is essential in gradient-based optimisation. AD is used extensively to obtain geometric gradients in this work, while FD is only used to compare gradients for the purpose of verification.

The computational cost of three methods introduced above (FD, complex-step, tangent linear) and forward mode AD scales linearly with the number of design variables. Therefore, the cost becomes prohibitive when dealing with optimisation of CFD involved systems with many design variables. As a consequence, an efficient method is need to compute expensive flow sensitivities, i.e. the adjoint method (see Section 4.3.1).

Chapter 4

CFD and in-house solvers

The past 20 years have witnessed a major increase in the use of CFD due to the continuous advances in computer technology. It has been used in a wide range of areas, such as aerospace, automotive, chemical engineering, turbo-machinery and shape optimisation. In this work, CFD is used to perform flow analysis within the CAD-based aerodynamic shape optimisation framework.

It is well known that the analytic solution to governing equations of flow problems (such as Navier-Stokes equations) are complicated, therefore only in a very small number of cases, such as some fully developed flows in simple geometries, is it possible to obtain an analytical solution [15]. In most cases, numerical methods are necessary. Several numerical algorithms are used in CFD, such as finite differences method (FDM), finite volume method (FVM), and finite element method (FEM). The FVM is our choice, and it will be briefly introduced in Section 4.1.

Two in-house CFD solvers based on FVM developed at the CFD group of QMUL are used in different test cases. The first one is an incompressible solver called GPDE, which utilises cell-centred FVM for spatial discretisation. The second solver STAMPS uses node-centred FVM for compressible flows. The adjoint solvers derived based on flow solvers are employed to provide cheap flow sensitivities, therefore the adjoint method as well as the adjoint solver, will also be introduced in this chapter.

Although solvers are key components of CFD-based shape optimisation framework, they are not the focus of this research. As will be demonstrated in Chapters 7, 8 and 9, the developed parametrisation method (see Chapter 5) is independent of solvers. The method can even work with structural solver to perform structural optimisation, provided the sensitivity is given. Therefore, the in-house solvers GPDE and STAMPS will only be

introduced briefly here. References will be provided where more details about them can be found if readers are interested.

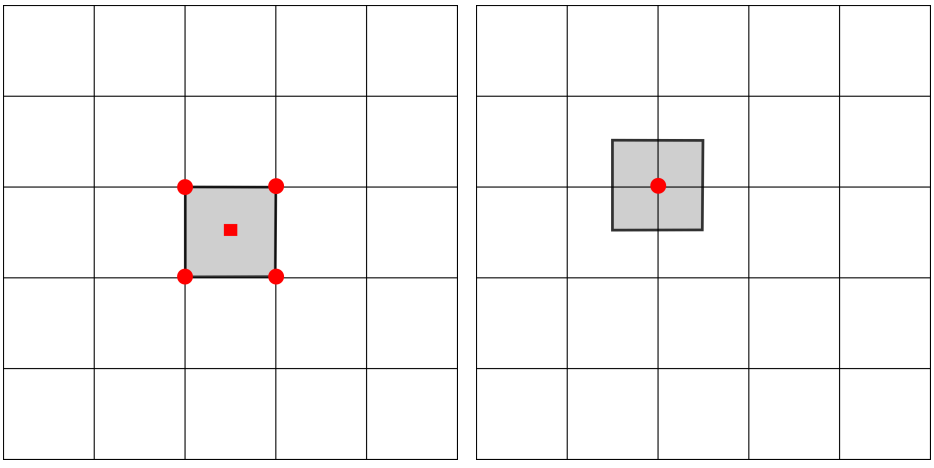
4.1 Finite volume method

The FVM is one of the numerical algorithms in CFD. It is chosen by many commercial CFD solvers, so do the in-house solvers used in this thesis.

In the FVM, the solution domain is subdivided into a finite number of small control volumes (CV). The governing equations (see equation (4.1)) are built for each cell and then summed over all the CVs.

Two basic approaches in FVM are:

- *Cell-centred scheme* (see Fig. 4.11(a)). In this approach, the CVs are identical to the grid cells, and flow quantities are stored at the centroid of grid cells. The in-house solver GPDE (see Section 4.2.1) is based on this method.
- *Node-centred scheme* (see Fig. 4.1(b)). In this method, the CVs are volumes centred around the grid vertices, where the flow quantities are stored. This is called dual CVs, on which the in-house solver STAMPS is based (see Section 4.2.2).



(a) Cell-centred scheme (b) Node-centred scheme

Figure 4.1: Control volume of cell-centred and node-centred scheme in FVM.

The generic conservative form of fluid flow equations can be written in the following

form [13]:

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho\phi \mathbf{u}_v) = \nabla \cdot (\Gamma grad\phi) + S_\phi, \tag{4.1}$$

where ϕ is a passive scalar, ρ is the density, t is the time, $\mathbf{u}_v$ is the velocity vector, Γ is the diffusivity coefficient, S_ϕ is the source term. Equation (4.1) is a *transport equation* for property ϕ. By associating ϕ with different variables, giving Γ different expressions and modifying S_ϕ accordingly, the continuity equation, momentum equation and energy equation can be obtained, as shown in Table 4.1.

Table 4.1: Value of different variables in the transport equation [13].

equation	ϕ	Γ	S_ϕ
continuity equation	1	0	0
x-momentum equation	u	μ	$-\frac{\partial p}{\partial x} + S_{M_x}$
y-momentum equation	v	μ	$-\frac{\partial p}{\partial y} + S_{M_y}$
z-momentum equation	w	μ	$-\frac{\partial p}{\partial z} + S_{M_z}$
energy equation	e	k	$-p\nabla \cdot \mathbf{u} + \Phi + S_i$

In Table 4.1, u, v, w are the velocity components in the x, y, z direction, respectively. Note that in this thesis, u and v are used to represent velocity components only in this Chapter. e is internal energy, μ is dynamic viscosity, k is thermal conductivity coefficient, p is pressure. S_{M_x}, S_{M_y}, S_{M_z} are momentum source in x, y, z direction, respectively. Φ is dissipation function, S_i is a source term. More details about this transport equation can be found in [13].

Equation (4.1) is the start point of FVM. One key step of the FVM is to integrate this equation over a 3-D CV, which yields:

$$\int_{CV} \frac{\partial(\rho\phi)}{\partial t} dV + \int_{CV} \nabla \cdot (\rho\phi \mathbf{u}_v) dV = \int_{CV} \nabla \cdot (\Gamma grad\phi) dV + \int_{CV} S_\phi dV, \tag{4.2}$$

where V is the volume of CV. According to the Gauss's divergence theorem [158], which states

$$\int_{CV} \nabla \cdot \mathbf{a} dV = \int_A \mathbf{n} \cdot \mathbf{a} dA, \tag{4.3}$$

where A is the boundary surface of control volume V, equation (4.2) can be rewritten as

$$\frac{\partial}{\partial t}\int_{CV} \rho\phi dV + \int_A \mathbf{n} \cdot (\rho\phi \mathbf{u}_v) dA = \int_A \mathbf{n} \cdot (\Gamma grad\phi) dA + \int_{CV} S_\phi dV. \tag{4.4}$$

Discrete equations are needed for computer to solve. In this case, the surface and volume integrals should be approximated. The simplest way to approximate the surface integrals is using mid-point rule, i.e. the product of integrand at the cell-face center and the cell-face area. The approximation of volume integrals can be obtained using the product of the mean value of the integrand and the CV volume. As a consequence, in the spatial discretisation, face value of variables are required. Various methods can be used to obtain the face value, such as the Upwind Differencing Scheme (UDS), Central Difference Scheme (CDS), Quadratic Upwind Interpolation (QUICK).

The integration conservation equation (4.4) is applied to every CV. If we sum equations for all the CVs, the conservation will be satisfied for the whole system. This is the reason why the global conservation is built into FVM, which is one of its principal advantages. One of other advantages of the FVM is that the spatial discretisation is carried out directly in the physical space, therefore there is no need to transform between the physical and computational space as in the FDM [159]. In addition, the FVM is very flexible such that in can be easily implemented on structured as well as on unstructured meshes. Because of these features, it is not surprising that FVM has been widely adopted by the majority of CFD software such as ANSYS Fluent [1], OpenForm [2], and SU2 [3], and also two in-house CFD solvers developed at QMUL.

More details on the FVM can refer to [13, 15, 160].

4.2 Flow solver

4.2.1 Incompressible flow solver: GPDE

GPDE is a solver for incompressible viscous steady flow based on unstructured grids and Navier-Stokes (N-S) equations. It is written in Fortran 90/95 programming language [161, 162]. It basically consists of two main components. The first one is an incompressible flow solver which solves the Navier-Stokes equations using FVM for spatial discretisation and SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) [15] algorithm for solving the discretised systems. The second component is a discrete adjoint solver (see Section 4.3.2) which computes the sensitivity of cost function w.r.t. surface mesh node coordinates, i.e. $\frac{\partial J}{\partial \mathbf{X}_s}$.

GPDE is based on the cell-centred FVM scheme, whose typical 2-D and 3-D Cartesian

[1] http://www.ansys.com/Products/Fluids/ANSYS-Fluent
[2] http://www.openfoam.com/
[3] http://su2.stanford.edu/

CVs are shown in Fig. 4.2. In these figures, the faces around a node (P) are represented by lower-case letters corresponding to the direction, such as w and e, while the nodes around are represented by upper-case letters, such as W and E.

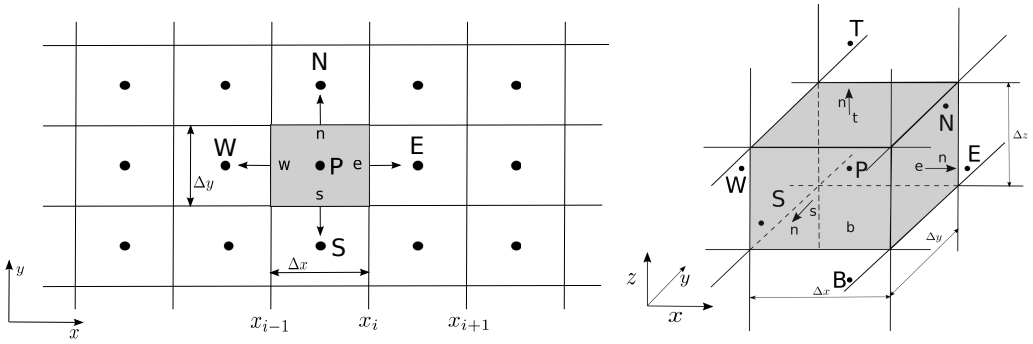


Figure 4.2: Typical 2-D (left) and 3-D (right) control volume in GPDE.

GPDE solves incompressible steady flow where the property is independent of time. Without density changes, the flow field can often be solved by only considering momentum and continuity equations. In addition, no body force is considered. Therefore, from equation (4.4) we can obtain the N-S equations need to be solved in GPDE:

$$\int_A \mathbf{n} \cdot (\rho \mathbf{u}_v \mathbf{u}_v) dA = \int_A \mathbf{n} \cdot (\mu \nabla \mathbf{u}_v) dA + \int_A \mathbf{n} \cdot (-\nabla p) dA, \tag{4.5}$$

$$\int_A \mathbf{n} \cdot (\rho \mathbf{u}_v) dA = 0, \tag{4.6}$$

where equation (4.5) is the momentum equation, (4.6) is the continuity equation.

As mentioned in Section 4.1, to solve these equations in computer, spatial discretisation is required to obtain discrete equations. The surface integrals need to be approximated, and in this case face value of velocity and pressure are required. In GPDE, both UDS and CDS are implemented, and the spatial gradients of flow field are computed by using Green-Gauss approach.

As can be seen from equation (4.5), the gradient of pressure is involved in the momentum equation as a part of the source term, and there is no explicit independent equation for pressure in the governing equations. Therefore, it is necessary to decouple velocity and pressure and update them to convergence. In GPDE, the SIMPLE algorithm [15] is utilised.

The basis process of SIMPLE algorithm is listed as following:

1. Guess a velocity field $\mathbf{u}_v^0$, use it to compute the coefficients in momentum equations
2. Guess a pressure field p^0, and solve the momentum equations to obtain more accurate velocity field $\mathbf{u}_v^*$
3. Solve the continuity equation using $\mathbf{u}_v^*$ to obtain pressure correction p', and update the pressure based on this correction value: $p^{new} = p^0 + p'$
4. Compute the velocity correction $\mathbf{u}_v'$ based on pressure correction obtained in last step, and update the velocity field: $\mathbf{u}_v^{new} = \mathbf{u}_v^* + \mathbf{u}_v'$
5. Solve all other discretised transport equations
6. Update the guess velocity field to $\mathbf{u}_v^{new}$, update the guess pressure field to p^{new}
7. Repeat step 1 - step 5, until convergence obtained.

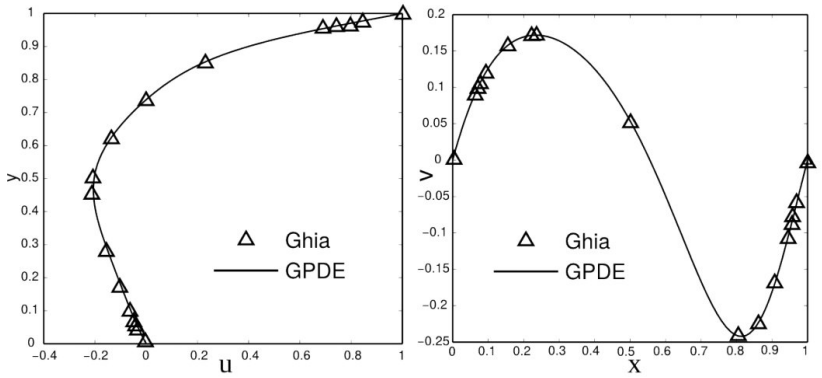
Through the SIMPLE algorithm, the N-S equations are decoupled into two sets of linear equations: discretised momentum and continuity (pressure correction) equations. There are basically two types of methods to solve linear equation systems, i.e. direct methods and iterative methods. Because of the high cost of direct methods, in GPDE, an iterative linear solver is chosen. To be more specific, in GPDE, conjugated gradient stabilized method (CGSTAB) and bi-conjugate gradient stabilized method (BI-CGSTAB) [163] are utilised to solve continuity equations and momentum equations, respectively.

GPDE also provides the mesh deformation technique, which is required in shape optimisation problem. In the shape optimisation framework utilised in this study as shown in Section 1.2, surface mesh deformation is performed using the in-house CAD kernel based on NURBS (see Chapter 5), while volume mesh deformation is performed in GPDE. Specifically, the spring analogy [164–166] is implemented. In this approach, each edge of the mesh is considered as a spring and the spring stiffness is taken as inversely proportional to the edge length.

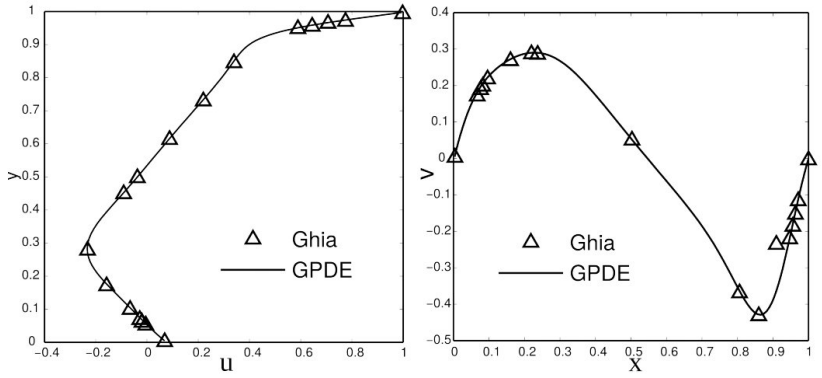
Validation of GPDE flow solver

Validation of flow solver is important in CFD simulations. The validation of GPDE flow solver has been performed by Wang [167], a former PhD student of the CFD group at QMUL and one of main developers of GPDE, for both 2D and 3D cases. Wang firstly ran the classic benchmark lid-driven cavity case, which is often used for validation of incompressible flow solvers, with GPDE and compared with reference value in the literature [168]. Figure 4.3 presents the quantitative comparison of GPDE results with

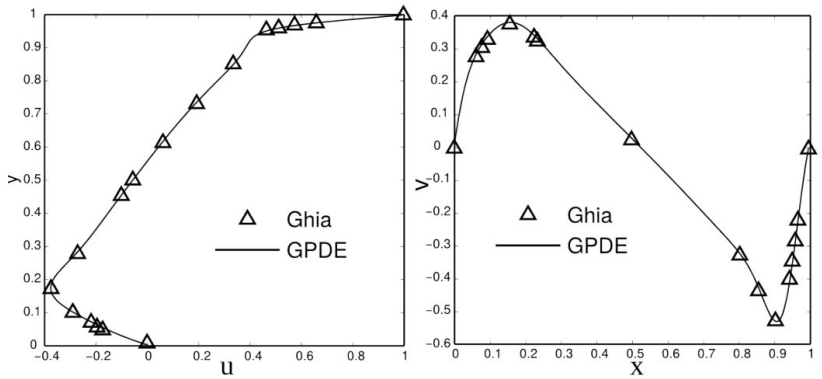
those provided by Ghia et al. [168], showing good agreement.



(a) Re=100, results for u velocity (left) and v velocity through the center of cavity



(b) Re=400, results for u velocity (left) and v velocity through the center of cavity



(c) Re=1000, results for u velocity (left) and v velocity through the center of cavity

Figure 4.3: Comparison of results for lid-driven cavity case between GPDE and the literature [167].

In this work, GPDE is applied to the shape optimisation of an S-bend air duct, which is a test case used in this study (see Chapter 7). In [167], Wang also carried out validation

using this case by comparing flow solutions given by GPDE flow solver with those of the widely used open-source CFD software OpenFOAM[4]. The mesh of this case is shown in Fig. 4.4, and more details of the S-bend case, please see Chapter 7. Quantitative comparison of velocity values over the same line in the flow solution is given in Fig. 4.5, which indicates clearly good agreements between GPDE and OpenFOAM results.

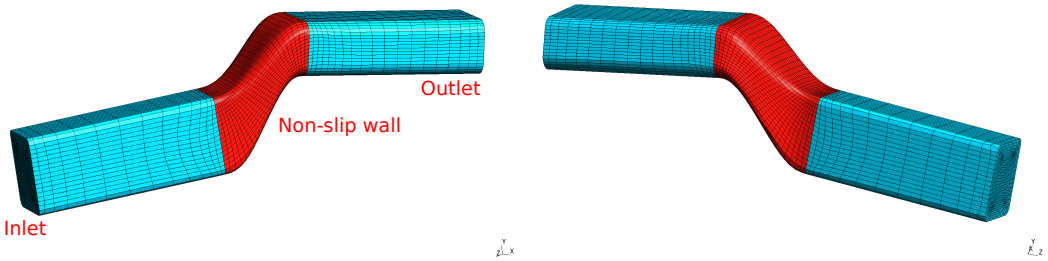


Figure 4.4: Hexahedral mesh of S-bend air duct from a VW Golf vehicle.

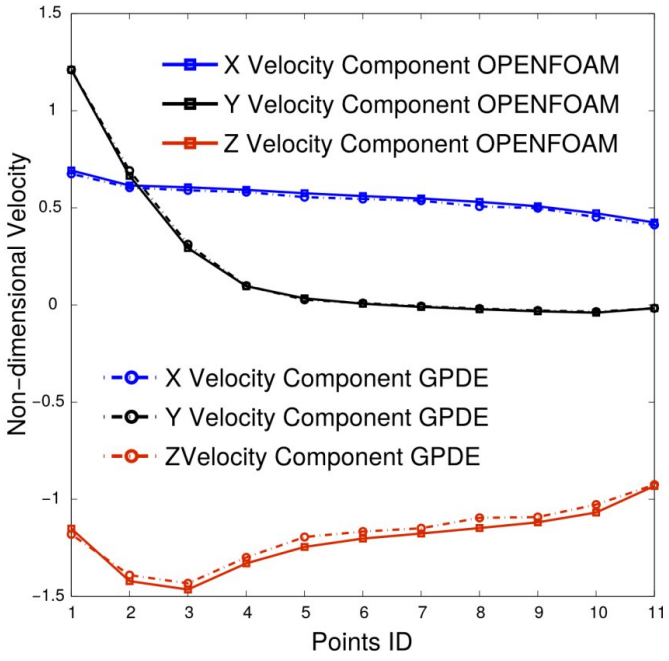


Figure 4.5: Comparison of velocity components obtained by GPDE and OpenFOAM [167].

The validation results for both 2D and 3D cases demonstrated that GPDE flow solver is reliable. This gives more confidence to any optimisation results obtained by including

[4]http://www.openfoam.com/

GPDE into the optimisation loop, such as the S-bend air duct shape optimisation which will be presented in Chapter 7.

The details about the implementation of GPDE are out the scope of this work, interested readers can find more details about the GPDE flow solver and also its validation from the work of GPDE developers [41, 167, 169].

4.2.2 Compressible solver: STAMPS

STAMPS (Source-Transformation Adjoint Multi-Physics Solver) [170] is an in-house compressible inviscid and viscous solver developed under several projects funded by European Commission, i.e. two previous projects FlowHead[5] and AboutFlow[6], as well as the ongoing IODA[7]. It contains flow solver and adjoint solver, and can provide shape sensitivity. In this work, it is used combining with NSPCC and optimiser to perform gradient-based shape optimisation.

STAMPS is a 3-D compressible solver solving the Reynolds-Averaged Navier-Stokes (RANS) on unstructured grids. The supported element types include tetrahedron, pyramid, prism and hexahedron.

The vertex-centred FVM (see Section 4.1) with an edge-based data structure is utilised in STAMPS, and dual CVs are constructed around grid nodes, where flow variables are stored. This differs from how GPDE works. Examples of dual CVs in STAMPS are illustrated in Fig. 4.6. Note that in this section, Ω denotes the volume of dual CV.

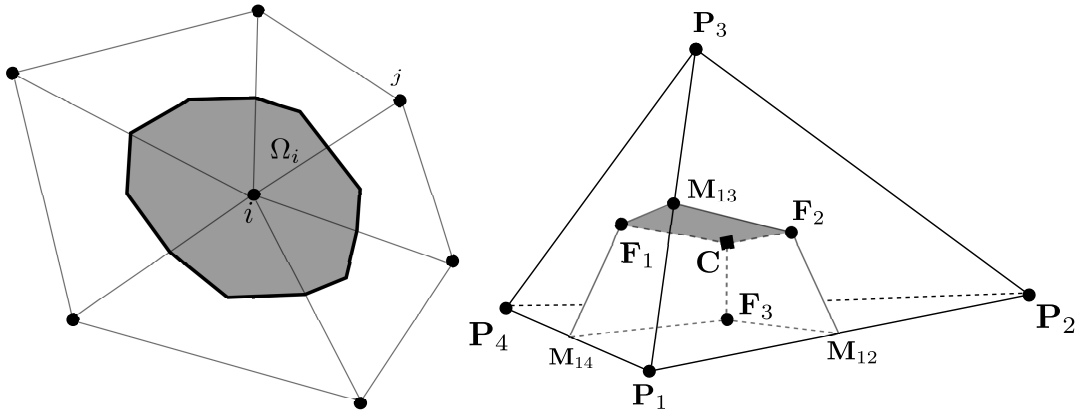


Figure 4.6: Left: a 2-D CV. Right: a partial 3-D CV, with **P** denotes grid nodes, **C** cell centroids, **M** edge midpoints, **F** face centres.

[5] http://flowhead.sems.qmul.ac.uk/
[6] http://aboutflow.sems.qmul.ac.uk/
[7] http://ioda.sems.qmul.ac.uk/

The RANS equations are presented as:

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{W}d\Omega+\oint_{\partial\Omega}(\mathbf{f}_c-\mathbf{f}_v)dS=\int_{\Omega}\mathbf{q}d\Omega \tag{4.7}$$

where Ω is the volume of the CV, $\partial\Omega$ the CV boundary, S the closed surface of CV. $\mathbf{W}$ is conservative variable vector, $\mathbf{f}_c$ and $\mathbf{f}_v$ are convective flux and viscous flux vector, respectively. $\mathbf{q}$ is the source term vector. The detailed expressions of $\mathbf{W}, \mathbf{f}_c, \mathbf{f}_v, \mathbf{q}$ are put in Appendix F to make the thesis compact. The Spalart-Allmaras turbulence model [171, 172] is used to close the RANS equation system.

In STAMPS, equation (4.7) is discretised with the method of lines, i.e. separate discretisation in space and in time.

Moving the source term in equation (4.7) to the left hand side, and using $\mathbf{R}$ to denote the residual which is a sum of convective fluxes, viscous fluxes and source terms at the CV Ω_i, yields:

$$\mathbf{R}(\mathbf{W})=\oint_{\partial\Omega_i}\mathbf{f}_c dS_i-\oint_{\partial\Omega_i}\mathbf{f}_v dS_i-\int_{\Omega_i}\mathbf{q}d\Omega_i, \tag{4.8}$$

which can then be approximated with the discretised integrals as:

$$\mathbf{R}(\mathbf{W})=\mathbf{F}_{c,i}-\mathbf{F}_{v,i}-\mathbf{Q}_i. \tag{4.9}$$

Note that STAMPS stores flow variables at mesh nodes, while the fluxes are computed at faces of CV. Therefore, the reconstruction of solution vectors are both sides of the face is needed.

In STAMPS, the convective flux $\mathbf{F}_{c,i}$ is computed by the ROE flux-difference splitting scheme [173] by default. The viscous flux $\mathbf{F}_{v,i}$ is calculated with an edge-based numerical integration. The volume source term $\mathbf{Q}_i$ is evaluated using the nodal value of the source term and the volume of CV Ω_i.

Regarding the temporal discretisation, both explicit and implicit solvers are implemented in STAMPS. Compared to the explicit solver, the implicit solver allows a much larger time step. By default, the Jacobian-Trained Implicit Runge-Kutta (JT-JIRK) [174] solver is used in STAMPS.

Different kinds of boundary conditions are supported in STAMPS, including:

- Far-field
- Subsonic inlet/outlet

- Inviscid (slip) wall
- Viscous (no-slip) wall
- Symmetry
- Periodic BC

Some of these boundary conditions will be utilised in test cases of this study in Chapters 8 and 9. In addition, the one-shot method [175, 176] is implemented to accelerate the convergence, which is also utilised in the optimisation test cases in Chapters 8 and 9.

Same as in GPDE, mesh deformation techniques are also contained in STAMPS. To be more specific, three methods are implemented, namely linear elasticity, spring analogy, and the inverse distance weighting (IDW) weighting [177].

Validation of STAMPS flow solver

The STAMPS flow solver is applied in test cases of this study to provide flow solutions, thus it is very important to make sure the solver is reliable. Validation of STAMPS has already been performed by its main developers, namely former PhD students of the CFD group at QMUL, Christakopoulos [61] and Xu [178], in their PhD theses.

In [178], the STAMPS flow solver was applied to solve the transonic turbulent flow around a 3D ONERA M6 wing [179]. The ONERA M6 wing is a well-known case for validation of compressible solver, since experimental results are available for comparison. In this case, main parameters are listed as following:

- Freestream Mach number $Ma = 0.84$
- Angle of attack (AoA) $= 3.06°$
- Reynolds number: 20 million

The geometry, surface pressure and Mach number contour are presented in Fig. 4.7. Note that for illustrative purpose, the wing is mirror about the symmetric plane in this figure.

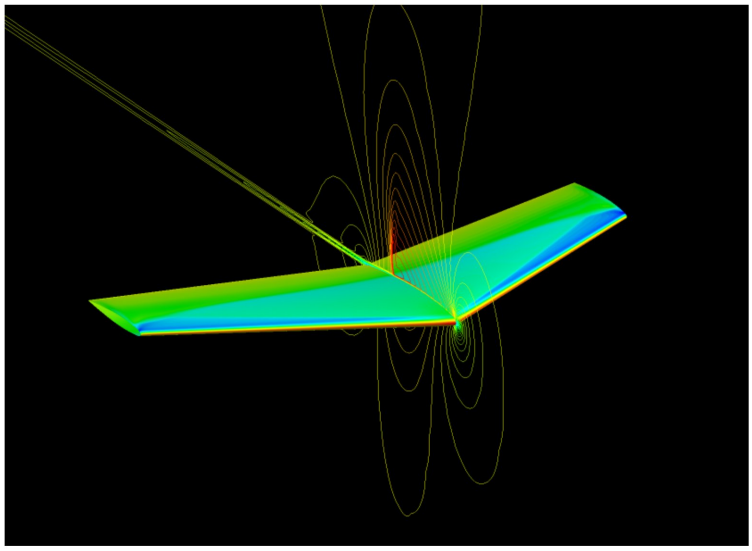


Figure 4.7: Overall geometry of ONERA M6 wing along with surface pressure distribution, and the Mach number contour isolines [178].

The STAMPS main developer Xu [178] also compared the solutions (pressure coefficient distribution) of M6 wing from STAMPS flow solver with those from Fluent Analysis, as well as experimental results provided by Schmitt and Charpin [179]. The comparison of pressure coefficient at different spanwise locations is illustrated in Fig. 4.8, showing good agreement among results. This validation indicates that STAMPS flow solver can provide reliable results for the ONERA M6 wing case.

In addition to turbulent flow over the ONERA M6 wing, another developer of STAMPS (previous mgOpt), Christakopoulos has validated the flow solver for inviscid transonic flow over the ONERA M6 wing [61]. It has been shown that even for inviscid computations, the results had a close match with experimental measurements. Christakopoulos demonstrated again that STAMPS can provide reliable results for ONERA M6 wing case.

Apart from the M6 wing case, the validation of STAMPS flow solvers have also been performed for other cases, such as the transonic turbulent flow over a 2D flat plate and transonic turbulent flow around RAE2822 airfoil. The interested readers are referred to [178].

The verification of the spatial discretisation accuracy of STAMPS was performed by another developer of STAMPS, Gugala, in his dissertation [180]. Their works have shown that STAMPS is a trustworthy solver. Interested readers could refer to those works for more details.

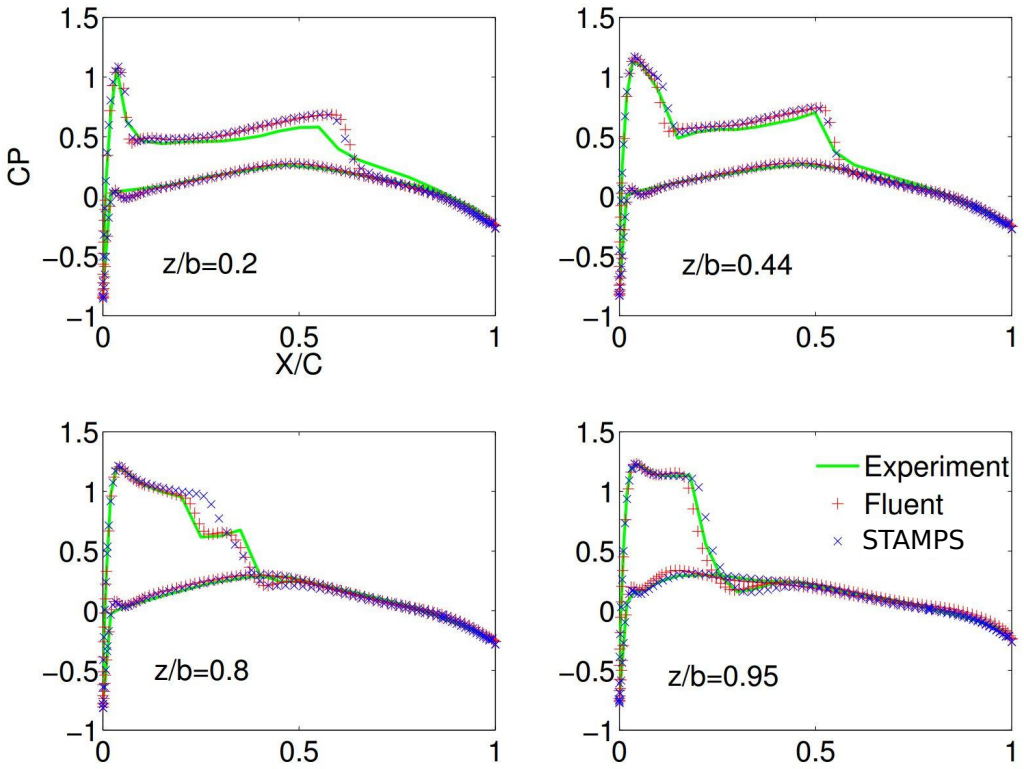


Figure 4.8: The comparison of pressure coefficient obtained from STAMPS, ANSYS Fluent and the experimental results [178].

4.3 Adjoint solver

As introduced in Section 3.4, efficient method is needed to compute the flow sensitivities w.r.t. design variables to avoid prohibitive computational costs. The adjoint method, which can provide gradients at a cost almost independent on the number of design variables, is the method of choice in the context of CFD-based optimisation, and will be introduced in this section. Also, both in-house solvers introduced above, namely GPDE and STAMPS, contain an adjoint solver. This section will also introduce these adjoint solvers briefly.

4.3.1 Adjoint Method

The adjoint method has been used in optimal control theory for a long time [181]. In 1974, Pironneau [37] introduced the adjoint approach to optimisation in fluid dynamics. In the area of aeronautical optimisation, it was firstly used by Jameson [38], and then used in

many other works [2, 21, 93, 182–187]. The main reason of using the adjoint method is that it is the only currently known method that can compute the gradient of the objective function with a computational cost that is nearly independent of the number of design variables [188]. Its total cost for the gradient computation depends on the language, the tool and the implementation. For example, STAMPS has a factor of 1.1 in runtime and memory, and SU2 has a factor of around 1.2 in time while 8 in memory [189].

There are different ways to derive the adjoint equations. We only present the derivation with Linear Algebra approach here, since it is most easily understood. Derivation of adjoint equations with different approaches could be found in [139].

The sensitivity of objective function J with respect to $\boldsymbol{\alpha}$ obtained in Section 3.4.3 can be formulated as:

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \frac{\partial J}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \mathbf{g}^T\mathbf{u}, \tag{4.10}$$

where $\mathbf{g}^T = \frac{\partial J}{\partial \mathbf{U}}$. In equation (4.10), the computation of the partial derivative $\frac{\partial J}{\partial \boldsymbol{\alpha}}$ is not expensive. Similarly, the source term $\mathbf{f}$ in equation (3.22) is computationally inexpensive, but depends on the design variable α_i. However, computing $\mathbf{u}$ (see equation (3.22)) involves a linear system solve for each component α_i of $\boldsymbol{\alpha}$, a cost which will become prohibitive if the number of design variables is large.

On the other hand, the sensitivity of the objective $\frac{\partial J}{\partial \boldsymbol{\alpha}}$ can be obtained without computing the perturbation field $\mathbf{u}$ with the adjoint method. Using the solution of equation (3.22), $\mathbf{u} = \mathbf{A}^{-1}\mathbf{f}$, in (4.10) and transposing equation (4.10) one obtains

$$\frac{dJ}{d\boldsymbol{\alpha}}^T = \frac{\partial J}{\partial \boldsymbol{\alpha}}^T + (\mathbf{g}^T\mathbf{A}^{-1}\mathbf{f})^T = \frac{\partial J}{\partial \boldsymbol{\alpha}}^T + \mathbf{f}^T\mathbf{A}^{-T}\mathbf{g}. \tag{4.11}$$

The right-most matrix-vector product $\mathbf{A}^{-T}\mathbf{g}$ can be interpreted as a linear equation similar to equation (3.22) for a new variable $\mathbf{v}$,

$$\mathbf{A}^T\mathbf{v} = \mathbf{g}. \tag{4.12}$$

With the solution for the adjoint variable $\mathbf{v}$ we can rewrite equation (4.11) as:

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \mathbf{v}^T\mathbf{f}, \tag{4.13}$$

where $\mathbf{v}$ is the adjoint variable. Based on $\mathbf{g}^T = \frac{\partial J}{\partial \mathbf{U}}$ and equations (3.22), the adjoint equivalence then states

$$\mathbf{g}^T\mathbf{u} = (\mathbf{A}^T\mathbf{v})^T\mathbf{u} = \mathbf{v}^T\mathbf{A}\mathbf{u} = \mathbf{v}^T\mathbf{f}, \tag{4.14}$$

which means the computation of second term on the right hand side of equation (3.23), can be achieved by either $\mathbf{g}^T\mathbf{u}$ or $\mathbf{v}^T\mathbf{f}$. At first glance, it seems that the computation cost of these two terms are similar. However, since the adjoint variables $\mathbf{v}$ depends only on the flow field through the transposed Jacobian $\mathbf{A}^T$ and on the objective through $\mathbf{g}$, equation (4.13) allows to compute the entire sensitivity vector $\frac{dJ}{d\boldsymbol{\alpha}}$ with a single system solver of equation (4.12). Hence, the cost of computing the gradient becomes almost independent of the number of design variables, but is proportional to the quantity of objective functions.

Since the computational cost of an adjoint linear solve is similar to the tangent linear case, for a system with one design variable and one cost function, the computational complexity of the adjoint method and tangent liner approach is similar, thus there would be no benefit in using the adjoint method. But for cases where there are more design variables than cost function, the adjoint methods will be computationally more efficient. Therefore, the adjoint method is very suitable for cases where design variables are much more than cost functions, which is usually the case in industry.

There are two kinds of adjoint approaches, namely the continuous adjoint method and the discrete one [182, 188]. The continuous adjoint method is to 'differentiate then discretise', which means the adjoint equations are formed starting from the flow equations and then discretised. The discrete method is to 'discretise then differentiate', the adjoint equations are obtained directly from differentiating the discretized flow equations [190]. Both types are summarised in Fig. 4.9.

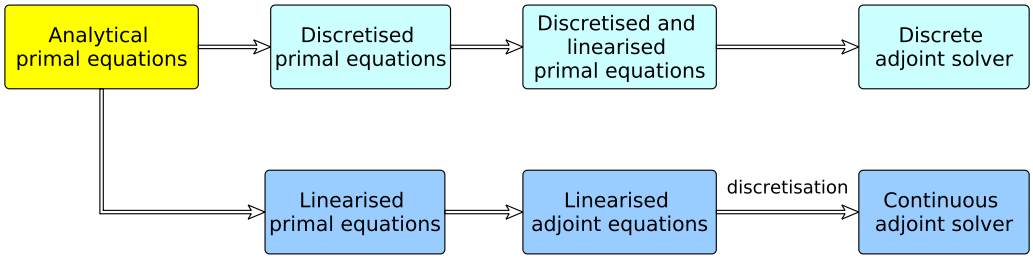


Figure 4.9: Discrete and continuous adjoint method.

Detailed comparison between the continuous and discrete adjoint method can be found in [40, 188, 190]. Generally speaking, the discrete adjoint method can provide exact discrete gradient of the objective function, which can then be rigorously verified using other methods, such as tangent linear. In addition, the discrete adjoint code derivation is straightforward conceptually. However, the hand-coded implementation of discrete adjoint method is cumbersome and error-prone, the code maintenance is also tricky. The

AD introduced in Section 3.4.4 is a promising way to address the implementation issue of the discrete method. Being able to use AD is also an advantage of the discrete method.

In this work, discrete adjoint solver is available in both in-house solvers GPDE and STAMPS to evaluate gradients. Both adjoint solvers are implemented with AD based on the source code of original flow solver.

4.3.2 Adjoint solver in GPDE

One of the main components in GPDE is a discrete adjoint solver, which is derived from the flow solver via AD with Tapenade. GPDE is written in Fortran 95, thus can be recognised by Tapenade. However, by applying AD to the whole flow solver in "brute-force" manner one may obtain very inefficient codes [191]. Therefore, AD is used to produce the relevant derivative source code for fluxes and source terms. The differentiated routines are then assembled in a hand-written driver code to improve performance.

The flow solver loop (primal loop) in GPDE for SIMPLE-family algorithms and its differentiated loop in reverse mode are listed in Listing 4.1 and Listing 4.2, respectively. U is flow solution vector, p is pressure, p‘ is pressure correction in the SIMPLE algorithm. As can be seen, the adjoint loop runs subroutines of the primal loop in reverse order.

```
1  Do i=1, n
2    call momentum_equation
3    call continuity_equation
4    if(last iteration) pushreal8array(U,p,...,p‘)
5  End do
6  call objective
```

Listing 4.1: Primal loop in GPDE [167].

```
1  call objective_b
2  Do i=1, n
3    if(last iteration) popreal8array(U,p,...,p‘)
4    call continuity_equation_b
5    call momentum_equation_b
6  end subroutine
```

Listing 4.2: The adjoint loop in reverse mode in GPDE [167].

There are several things should be noted when using AD to derive the discrete adjoint solver. Firstly, the codes may contain features that cannot be parsed by the AD tool, this is because the AD tool is still under development. Therefore, the source code may need to be adjusted based on the tool's capabilities. Secondly, as mentioned before, it

is better to only differentiate the necessary parts. Thus, the codes need to be selected and then provided to Tapenade. Thirdly, to have better performance one should hand assemble the sensitivity code once, this requires good understanding of the underlying operations and needs user efforts. Fourthly, the differentiated subroutine may also need to be modified by hand to improve the performance and reduce the runtime. One reason of this is that, the Tapenade is based on S-T, thus it will firstly copy the codes in primal loop and then produce the differentiated codes. This results in redundant codes in the differentiated continuity equation shown in Listing 4.2. Finally, once previous steps are finished, a Makefile can be used to automate the process of differentiation, linking and compiling. The general process of deriving the adjoint solver is summarised in Fig. 4.10.

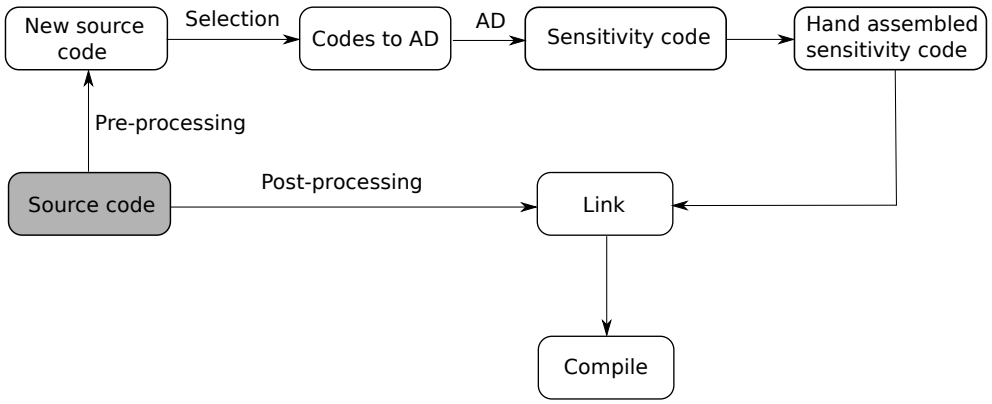


Figure 4.10: The general process of deriving adjoint solver using AD.

The general running process of GPDE is given in Algorithm 4.1. When using GPDE, the user needs to specify the running mode, namely primal (flow solver) or adjoint mode. The adjoint mode will provide the sensitivity $\frac{dJ}{d\mathbf{X}_s}$ which is needed in shape optimisation.

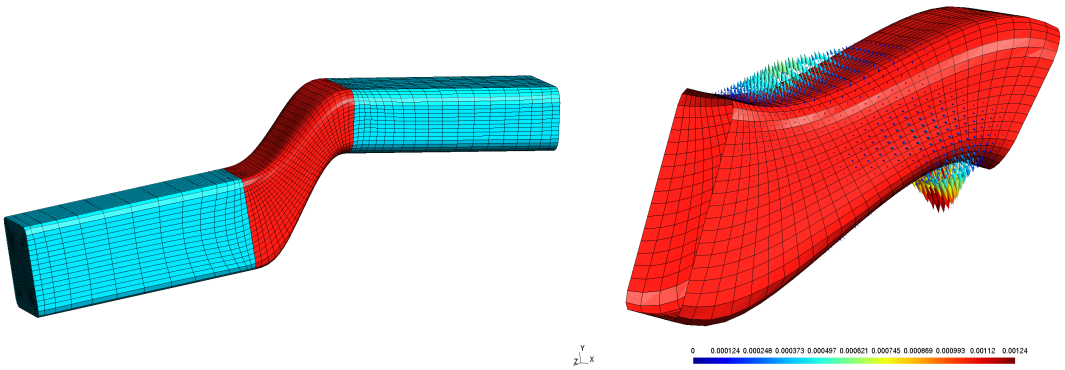
Once the adjoint solver is available and integrated into the gradient-based optimisation framework, the gradient should be verified before performing optimisation. The comparison of gradients computed from GPDE versus those from FD will be presented in Section 7.4, where GPDE is applied to a practical S-bend air duct shape optimisation problem.

Figure 4.11 presents the design surfaces of the S-bend air duct and normal sensitivity vectors computed from GPDE adjoint solver. These vectors point the direction where the objective function can be reduced. In this case, the objective function is the total pressure drop. As can be seen that most vectors point outwards, indicating the volume should be enlarged, which is physically valid. Note that here the design variables are the

Algorithm 4.1: Running process in GPDE

```
Given surface mesh coordinates X_s;
Given volume mesh;
Given the setting JSON file, including boundary conditions, Reynolds number,
  etc.;
switch Primal do
    Run mesh deformation subroutines;
    Run the subroutines solving flow governing equations ;
    Compute the cost function J;
end
switch Adjoint do
    Run Primal;
    Run differentiated flow subroutines ;
    Run differentiated mesh deformation subroutines;
    Compute the sensitivity of cost function w.r.t. mesh node coordinates, dJ/dX_s;
end
```

surface mesh node coordinates, while in Chapter 7 the CAD-related parameters are used as design variables.



(a) Design surfaces (shown in red) (b) Normal sensitivity

Figure 4.11: Normal sensitivity vectors of an S-bend air duct.

The details on the implementation of discrete adjoint solver in GPDE, and more verification results can be found in [167].

4.3.3 Adjoint solver in STAMPS

As introduced in Section 4.2.2, STAMPS contains a discrete adjoint solver which can provide the sensitivity of cost function w.r.t. surface mesh coordinates. The derivation

of this adjoint solver is similar to that of GPDE in following aspects:

- The AD tool Tapenade is utilised to produce sensitivity code.
- The AD is not applied to the entire flow solver, but to selected subroutines. The differentiated subroutines are then used to form the adjoint solver manually.
- Source code pre-processing and post-processing are also needed.

In other words, the process shown in Fig. 4.10 also applies to the derivation of the adjoint solver in STAMPS.

Equation (4.12) can be rewritten as:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \frac{\partial J}{\partial \mathbf{R}}^T = \frac{\partial J}{\partial \mathbf{U}}^T . \tag{4.15}$$

Then define

$$R_{ADJ} = \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \frac{\partial J}{\partial \mathbf{R}}^T - \frac{\partial J}{\partial \mathbf{U}}^T . \tag{4.16}$$

In STAMPS, the residual and objective subroutines are carefully prepared to be differentiated by Tapenade, to obtain $\frac{\partial \mathbf{R}}{\partial \mathbf{U}}$ and $\frac{\partial J}{\partial \mathbf{U}}$, respectively. The differentiated subroutines are then assembled in the adjoint solver, as illustrated in Listing 4.3, where one iteration of adjoint system is shown. The superscript down arrow $^{\downarrow}$ and up arrow $^{\uparrow}$ denote the input and output, respectively. U is flow solution vector, X is mesh coordinate, J is cost function, R is residual. The suffix 'b' denotes reverse mode.

```
1   Jb = 1.0
2   call objective_b_u(X↓, U↓, Ub↑, J↑, Jb↓)
3
4   solver_b:
5     R_ADJ = -Ub
6     Rb = v^n
7
8     call residual_b_u(..., U↓, Ub↑, R↑, Rb↓)
9     R_ADJ = R_ADJ + Ub
10
11    δv^n = linearSolver(..., R_ADJ)
12    v^{n+1} = v^n + δv^n
```

Listing 4.3: The adjoint solver pseudo code in STAMPS [180].

In addition, another subroutine is defined and differentiated in STAMPS to obtain the sensitivity of the objective function w.r.t. mesh coordinates, $\frac{dJ}{d\mathbf{X}}$, which can be derived

from equations (3.21), (4.13) and (4.14) as:

$$\frac{dJ}{d\mathbf{X}}^T = \frac{\partial J}{\partial \mathbf{X}}^T - \frac{\partial \mathbf{R}}{\partial \mathbf{X}}^T \frac{\partial J}{\partial \mathbf{R}}^T . \tag{4.17}$$

The pseudo-code to compute $\frac{dJ}{d\mathbf{X}}^T$ is:

```
1  Jb = 1.0
2  Rb = -v
3  call RL_b_x(X↓, Xb↑, U_init↓, U↓, R↑, Rb↓, J↑, Jb↓)
4  DLDX = Xb
```

Listing 4.4: The sensitivity assembly part in STAMPS [180].

This subroutine only needs to be called once to obtain the sensitivity of the cost function w.r.t. all mesh coordinates. After obtaining $\frac{dJ}{d\mathbf{X}}$, one can obtain the sensitivity w.r.t. surface mesh coordinates $\frac{dJ}{d\mathbf{X}_s}$ through the differentiated mesh deformation algorithm. $\frac{dJ}{d\mathbf{X}_s}$ is then needed in shape optimisation as introduced in Section 3.2.

In this study, the ONERA M6 wing[8] is employed for verification of sensitivities from the adjoint solver of STAMPS. The ONERA M6 wing is also utilised as a test case for shape optimisation in Chapter 9, where more details on this case can be found.

Case informations for verification are listed as following:

- Mesh: tetrahedral mesh with 135,204 nodes
- Flow conditions:
  - Freestream Mach number $Ma = 0.84$
  - Freestream temperature $T_\infty = 300K$
  - Angle of attack (AoA) $= 3.06°$
  - Pressure $p_\infty = 101325Pa$
- Objective function: lift

Several mesh nodes are selected for verification, and the sensitivities $\frac{dJ}{d\mathbf{X}_s}$ computed from the adjoint solver in STAMPS are compared with those from central finite differences (step size: 0.0001), as shown in Table 4.2. As can be seen, the results from both methods have good agreement, indicating that the adjoint solver in STAMPS is reliable.

[8] http://www.onera.fr/en

Table 4.2: Verification of sensitivities from STAMPS.

Mesh point	STAMPS	FD	Difference
122165	$-$**14.427**781261474593	$-$**14.427**870000000000	8.87385×10^{-5}
126197	**4.699**596875070711	**4.699**2721000000000	3.24775×10^{-4}
126204	$-$**6.422**5989626720548	$-$**6.422**420000000000	-1.78962×10^{-4}
126245	**17.2850**60484937276	**17.2850**00000000000	6.04849×10^{-6}
126247	**8.3736**246688204119	**8.3736**09999999999	1.46688×10^{-5}
126257	**7.7243**006307980977	**7.7243**10000000000	9.36920×10^{6}
126267	**7.8689**888323424384	**7.8689**00000000000	8.88323×10^{-5}

More details on the discrete adjoint solver in STAMPS can refer to [42, 61, 178, 180]. To be more specific, Xu [178] and Gugala [180] introduced mainly the underlying principle of deriving the adjoint solver via AD, while Christakopoulos [61] provided more details on the implementation, such as the source code preparation and post-processing after running Tapenade.

4.4 Summary of solvers

4.4.1 Summary of in-house solvers

In previous sections, both in-house solvers GPDE and STAMPS are introduced briefly, including the flow solver and adjoint solver components. There are some common features between these two solvers and some of them as listed as following:

- Contain flow solver and discrete adjoint solver
- Based on the FVM
- Written in Fortran 90/95 programming language
- The adjoint solver is derived via the AD tool Tapenade
- Support the GMSH[9] format

However, there are much more differences between them, some of which are summarised as following:

- GPDE is an incompressible solver, while STAMPS is a compressible solver, thus the equations solved are different

[9] http://gmsh.info/

- GPDE uses cell-centred scheme, while STAMPS is based on vertex-centred method with dual control volume
- GPDE solves steady problem, while STAMPS can handle both steady and unsteady flow

The present research focuses on CAD-based parametrisation and shape optimisation. Both in-house solvers GPDE and STAMPS are utilised in this work to provide flow solution and shape sensitivity based on adjoint method, such that CAD-based shape optimisation can be performed. The interfaces between these two solvers and other components in the shape optimisation framework are different. GPDE stores the sensitivity of the objective function w.r.t. surface mesh coordinates in an ASCII file which should be read by the optimisation script. In addition, GPDE reads the surface mesh perturbation also from an ASCII file which is produced by an in-house CAD kernel (see Chapter 5). In contrast, STAMPS utilises the HDF5 format[10] to store results, such as flow variables, sensitivity and mesh perturbation. This makes it easier to handle the 'heavy data' in CFD and adjoint. In this study, optimisation scripts have been developed to effectively interact with both solvers and perform the optimisation shown in Fig. 1.3 automatically. It shall be mentioned that the Python script used to read and write HDF5 files are provided by the author's colleague Rejish Jesudasan.

Note that in the present work, some assumptions and simplifications are made to flow solvers, or in other words, are made to optimisation test cases. For example, some of them are listed as following:

- Only single-phase flow is considered
- No heat transfer is considered
- Only the U-bend test cases (Chapter 8) considered is turbulent
- Supersonic flow is not considered at the moment

4.4.2 Other solvers

As will be demonstrated in later chapters, both GPDE and STAMPS work well within the shape optimisation framework with the developed NURBS-based parametrisation approach. Being able to work fine with so different two solvers clearly demonstrate the effectiveness of parametrisation method, and the modularity of CAD-based optimisation

[10] https://support.hdfgroup.org/HDF5/

framework. Besides, in [49], Xu et al. demonstrated that the NSPCC approach worked well with the Rolls-Royce in-house code HYDRA. Clearly, the framework is created modular so that other CFD solvers can be utilised in the place of GPDE and STAMPS, i.e. using a different solver does not affect the layout of the framework.

However, it is worthwhile giving a discussion on other widely used solvers, in relation to CAD-based geometry parametrisation and shape optimisation.

4.4.2.1 Commercial solvers

ANSYS Fluent[11] is the most famous and widely used commercial CFD tool. It contains the broad physical modelling capabilities to model flow, turbulence, heat transfer, combustion, etc.

Fluent has been extensively used in flow simulations, but to perform gradient-based optimisation, the gradient of objective function w.r.t. mesh coordinates are necessary. The adjoint method enables the inexpensive gradient computation available. The implementation of adjoint solver needs source code of flow solver, no matter to implement the continuous adjoint solver or discrete one. However, the source code of commercial solvers, such as Fluent, are not available for users. Although Fluent developed adjoint codes, they appear to have severe limitations in robustness and capability [192].

Because of above reasons, normally gradient-free methods are used as optimiser in shape optimisation when using Fluent as the flow solver, for example the Genetic Algorithm (GA) [84, 120, 121, 193] and evolutionary algorithm [194]. There are only very few studies performing aerodynamic optimisation with the adjoint solver of Fluent [195, 196].

Another famous commercial solver is STARCCM+, which shares the same problem as ANSYS Fluent when applied to shape optimisation problems. Therefore, to date only very few studies have been performed with STARCCM+ adjoint solver [197].

4.4.2.2 Open source solvers

Apart from commercial solvers, there are also open-source solvers which are widely used, such as the incompressible OpenFOAM[12][198], and the compressible SU2[13][199–202].

OpenFOAM

[11] https://www.ansys.com/Products/Fluids/ANSYS-Fluent
[12] https://www.openfoam.com
[13] https://su2code.github.io

OpenFOAM is a free, open source CFD software, and has extensive range of features to make it applicable for many problems, such as complex fluid involving chemical reactions, turbulence and hear transfer, etc. An adjoint solver is also available for steady-state RANS, as a result, OpenFOAM have been widely used in various shape optimisation problems [7, 20, 94, 203, 204].

In [94], OpenFOAM was utilised to provide shape sensitivity in an S-bend duct optimisation where the objective is to minimize the dissipated power. In that work, an explicit-defined CAD parametrisation (see Section 2.3.2.1) using CATIA V5 was employed. Garcia et al. [20] used OpenFOAM in wing shape optimisation, and a B-Spline curve is chosen to describe the airfoil. In [7], OpenFOAM was applied to automotive industry to perform both topology optimisation and shape optimisation ducted flows. Othmer et al. [203] proposed a method to translate the surface sensitivities obtained from OpenFOAM adjoint code to improved shape. Verstraete et al. [204] employed OpenFOAM to predict fluid dynamics and hear transfer performance in a U-bend cooling channel.

SU2

SU2 is an open-source collection of software tools for the analysis of partial differential equations, and is a leading technology for adjoint-based optimisation, from Stanford University. The core of SU2 is a RANS solver which is capable of solving the compressible, turbulent flow based on unstructured meshes. Besides, SU2 also contains an adjoint solver, therefore gradient information is available for optimal shape design, uncertainty quantification, etc. Indeed, SU2 has recently been developed to a software suite for solving complex, multi-disciplinary problems. One of the key features of SU2 is that it is designed for specific functionality and can be executed individually. Consequently, users can couple multiple modules and integrate with optimisers [205]. SU2 has been employed in many shape optimisation studies in the research community, such as the optimisation of airfoils [74, 205–208] and aircraft [209]. For more information about SU2, please refer to [199–202].

OpenFOAM and SU2 are good solvers, and have been rigorously tested. However, both OpenFOAM and SU2 mainly use the continuous adjoint method, although they also have discrete variants derived using operator overloading (see Section 3.4.4). For modelling beyond RANS, the continuous adjoint method suffers from difficulties with stability, and needs substantial efforts to derive the adjoint solver [170]. What's more, codes like OpenFOAM and SU2 are written in C++ and cannot be differentiated easily by source transformation method (S-T). The method used for them is operator overloading (O-O), whose memory requirements are substantial thus not appropriate for industrial

or large-scale computation on typically available memory.

It should be mentioned that, normally in research groups, to develop methods or algorithms, in-house codes or open source solvers are used. One of the main focus of the CFD group at QMUL is adjoint differentiation by application of source transformation AD tool (Tapenade) to CFD solvers. However, for advanced object-oriented languages such as C++, the AD tool using S-T are not currently available. Because of this, the CFD codes in our group (STAMPS and GPDE) are written in Fortran to be parsed and differentiated by Tapenade. As a member of this research group, in the present study the author used Tapenade and these solvers (STAMPS, GPDE).

Chapter 5

NURBS-based parametrisation with continuity constraints

The CFD optimisation group at QMUL has developed an in-house CAD kernel termed NURBS-based parametrisation with continuity constraints (NSPCC) [47–49]. This chapter presents further investigation and developments performed during this research, which aims to add understanding to the approach and make it more powerful and automatic.

5.1 Introduction to NSPCC approach

NSPCC is an in-house CAD kernel based on BRep, which is the most commonly used approach to define a geometry in CAD system as introduced in Section 1.3. In BRep, the skin of geometry is defined by surface patches which are connected together. The main objective of NSPCC is to provide a CAD engine which can provide exact and robust derivatives, such that it can be integrated into the design loop to make the efficient and automatic gradient-based shape optimisation feasible. The main feature of NSPCC compared to other approaches [50, 105] is that, it can deal with geometric continuity constraints between patches, therefore the application is not limited to geometries which contain one single surface.

The framework of CAD-based shape optimisation, which integrates the CAD kernel and solvers (flow and adjoint solver) is shown in Fig. 1.3. For convenience, here we extract the NSPCC part from the framework and present it in Fig. 5.1. NSPCC consists of five modules, namely LOADSTEP, FINDPARAMS, SURFDERIVS, PERTURBXS,

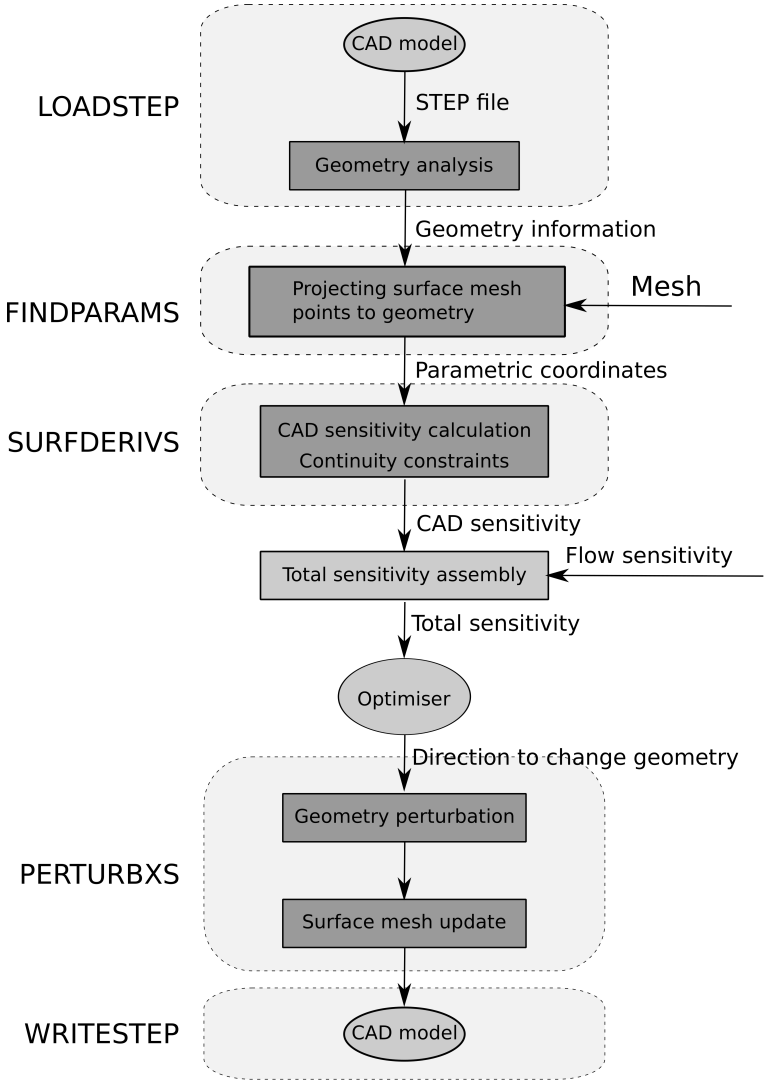


Figure 5.1: The work flow of NSPCC and name of each module

.

WRITESTEP. Each of them has different functionality.

The first module is the LOADSTEP module, which is used to read and parse a STEP file exported by CAD software, such as CATIA V5 and Solidworks, or written by hand. The LOADSTEP module extracts geometry and topology information from the STEP file, i.e. the information about surfaces and curves, as well as how surfaces are connected with each other. This module is the starting point of the CAD-based shape optimisation framework. More details on the STEP format will be discussed in Section 5.8.

The second module is FINDPARAMS which performs mesh projection, i.e. matching

the CAD geometry and surface mesh to find out the surface on which each mesh node lies. This process of mapping a point in 3D Cartesian coordinate system to a 2D parametric coordinate system, i.e. from (x, y, z) to (u, v) is referred as the *point inversion problem* [51, 210, 211]. This process is the link between CAD shape and the computational grids, thus is essential for CAD-based shape optimisation. The parametric coordinates (u, v) found in this module will be used in following modules. Details of point inversion will be discussed in Section 5.3.

The third module, SURFDERIVS, is used to handle geometric continuity constraints and compute CAD sensitivities (see Section 5.4.6). These geometric derivatives and flow sensitivities will be combined together and provided to optimiser. The assembled total sensitivities point out a direction along which the design can be improved.

The PERTURBXS module perturbs the geometry according to the output of optimiser. Normally, this perturbation can improve the design to some extent. Then, the surface mesh grids will be updated based on the new CAD surface.

Finally, the WRITESTEP module writes the updated geometry into a new STEP file, which can be handled by the LOADSTEP module and CAD software, as well as many other engineering software, such as Gmsh[1] and Ansys[2]. The WRITESTEP module is the end of the CAD-based shape optimisation framework.

5.2 Extension of in-house CAD kernel to support NURBS

Due to the numerous popularity of NURBS, in this work we extend the in-house CAD kernel NSPCC from B-splines to NURBS. This consists of two aspects. Firstly, the CAD kernel should be able to extract information about NURBS geometry from the STEP file and to write updated information back into the STEP file. Secondly, the CAD kernel should be able to include weights of NURBS control points in the design space, apart from the positions. The reading and writing of STEP file will be discussed in Section 5.8, here we discuss how to include weights in the design space.

As can be seen from the definition of NURBS (equation (2.1)), three factors have impact on the shape of a NURBS surface given the basis functions, namely position of the control point, weight of the control point and the knot vector. In [48, 49, 178], only the positions of control points are used. In this study, the approach is extended to NURBS, i.e. weights can also be perturbed by using the homogeneous form of NURBS. For con-

[1] http://gmsh.info/
[2] http://www.ansys.com/

venience, the homogeneous form of a NURBS surface is recalled in equation (5.1).

$$\mathbf{S}^{\omega}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}^{\omega}_{i,j} \qquad 0 \leq u,v \leq 1, \tag{5.1}$$

where $\mathbf{P}^{\omega}_{i,j} = (\omega_{i,j}x_{i,j}, \omega_{i,j}y_{i,j}, \omega_{i,j}z_{i,j}, \omega_{i,j})$.

A B-spline surface is expressed as [51]

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}_{i,j} \qquad 0 \leq u,v \leq 1. \tag{5.2}$$

By comparing equations (5.1) and (5.2) one can see that, if written in homogeneous form, a NURBS surface has the same form as a B-spline surface. Therefore, the homogeneous form allows one to work with nonrational polynomials, and most algorithms formulated for B-spline surfaces, such as the computation of surface point, can straightforwardly be applied to NURBS surface. In this way, the weights of control point are included in the design space.

In NSPCC, we firstly do the computations using the homogeneous of NURBS surface in 4-D space, such as the perturbation of homogeneous control points, then project the NURBS surface to 3-D space. The feasibility of the homogeneous coordinates is ensured by the invariant property of NURBS under projective transformation [60]. The homogeneous form of NURBS surface has been applied in structural optimisation problem recently in [28–30]. In the present work, we apply the homogeneous from in aerodynamic shape optimisation. To the best of the author's knowledge, the present work is the first to apply homogeneous form of NURBS in CFD-based shape optimisation with gradient-based optimiser. In addition, continuity constraints are handled while those work [28–30] did not. More details on the homogeneous form of NURBS surface can be found in [51, 52, 212, 213].

For simplicity, we will omit the superscript ω and use $\mathbf{P}$ for the homogeneous control point coordinate throughout the following sections, unless noted otherwise.

5.3 Mesh projection

In shape optimisation problem, although mesh is generated based on the CAD geometry initially, the discrete surface grid and the NURBS representation of the geometry are actually independent in the program. The program has no idea how to update the mesh

if the geometry is perturbed. Therefore, a link between them is required such that the mesh can be deformed along with the perturbation of geometry automatically. This is achieved by projecting the mesh nodes onto the NURBS patches and assigning the corresponding parametric coordinates (u, v) to each surface mesh points. In this way, during the optimisation process, the surface mesh points can be easily mapped onto the deformed NURBS surface to obtain new spatial coordinates with the parametric coordinates (u, v). This guarantees that the surface grid points always remain on the NURBS surface.

Good performance of mesh projection is crucial for the shape optimisation problem. An example of the mesh projection and deformation process is illustrated in Fig. 5.2.

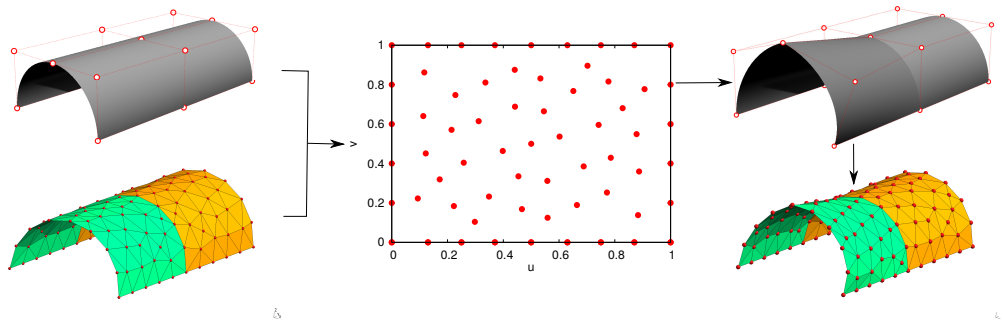


Figure 5.2: The mesh projection of half-cylinder and the deformation process.

The point inversion algorithm is devoted to perform mesh projection in this work, to find the parametric coordinates (u, v) satisfying $\mathbf{S}(u, v) = \mathbf{X}_s$, where $\mathbf{X}_s$ is a mesh node, $\mathbf{S}$ is a NURBS surface (see equation (2.1)). Note that the parametric coordinates (u, v) are maintained throughout the optimisation once found.

Piegl et al. [51] pointed out that it is not generally possible to find a closed form solution of the point inversion problem, thus the available algorithms are all iterative methods [51, 105, 210, 211, 214–216]. The method proposed by Piegl et al. [51], which finds the approximated solution by minimising the Euclidean distance between the mesh node and NURBS surface, is utilised in NSPCC. The condition to find the minimum distance is that the vector pointing from $\mathbf{X}_s$ to a point on the surface is orthogonal to the tangent vector at that point, which can be described as

$$\begin{cases} f(u, v) = (\mathbf{S}(u, v) - \mathbf{X}_s) \cdot \mathbf{S}_u(u, v) = 0, \\ g(u, v) = (\mathbf{S}(u, v) - \mathbf{X}_s) \cdot \mathbf{S}_v(u, v) = 0, \end{cases} \tag{5.3}$$

where $\mathbf{S}_u$ and $\mathbf{S}_v$ are the first partial derivative of surface point with respect to u and v,

respectively.

The Newton iteration starting with an initial guess (u_0, v_0) can be used to find the roots of $f(u, v)$ and $g(u, v)$, which is

$$\begin{bmatrix} u_{i+1} \\ v_{i+1} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix} - \mathbf{J}_i^{-1} \boldsymbol{\kappa}_i, \tag{5.4}$$

where i and $i+1$ indicate different iteration, and

$$\mathbf{J}_i = \begin{bmatrix} f_u & f_v \\ g_u & g_v \end{bmatrix} = \begin{bmatrix} \mathbf{S}_u \cdot \mathbf{S}_u + (\mathbf{S} - \mathbf{X}_s) \cdot \mathbf{S}_{uu} & \mathbf{S}_u \cdot \mathbf{S}_v + (\mathbf{S} - \mathbf{X}_s) \cdot \mathbf{S}_{uv} \\ \mathbf{S}_u \cdot \mathbf{S}_v + (\mathbf{S} - \mathbf{X}_s) \cdot \mathbf{S}_{vu} & \mathbf{S}_v \cdot \mathbf{S}_v + (\mathbf{S} - \mathbf{X}_s) \cdot \mathbf{S}_{vv} \end{bmatrix}, \tag{5.5}$$

$$\boldsymbol{\kappa}_i = \begin{bmatrix} f(u_i, v_i) \\ g(u_i, v_i) \end{bmatrix}, \tag{5.6}$$

where all the functions in the matrix $\mathbf{J}_i$ and $\boldsymbol{\kappa}_i$ are evaluated at (u_i, v_i), $\mathbf{S}_{uu}$, $\mathbf{S}_{uv}$, $\mathbf{S}_{vv}$ are the second derivatives of NURBS surface point w.r.t. parametric coordinates.

The convergence criteria are given by:

$$\begin{cases} |(u_{i+1} - u_i)\mathbf{S}_u(u_i, v_i) + (v_{i+1} - v_i)\mathbf{S}_v(u_i, v_i)| \leq \epsilon_1 \\ |\mathbf{S}(u_i, v_i) - \mathbf{X}_s| \leq \epsilon_1 \\ \dfrac{|\mathbf{S}_u(u_i, v_i) \cdot (\mathbf{S}(u_i, v_i) - \mathbf{X}_s)|}{|\mathbf{S}_u(u_i, v_i)||\mathbf{S}(u_i, v_i) - \mathbf{X}_s|} \leq \epsilon_2 \\ \dfrac{|\mathbf{S}_v(u_i, v_i) \cdot (\mathbf{S}(u_i, v_i) - \mathbf{X}_s)|}{|\mathbf{S}_v(u_i, v_i)||\mathbf{S}(u_i, v_i) - \mathbf{X}_s|} \leq \epsilon_2 \end{cases} \tag{5.7}$$

where ϵ_1 is a measure of Euclidean distance, ϵ_2 is a zero cosine measure. More details on this point inversion algorithm are omitted here, interested readers can refer to [51].

The performance of the Newton iteration introduced above is closely related to the choice of initial value (u_0, v_0). In this study, the initial value of the Newton iteration is setted by evenly choosing (u, v) values on the surface. To be more specific, a patch is divided into $N \times N$ small patches, leading to $(N+1) \times (N+1)$ grids. The parametric coordinates of these grids can be obtained from the initial information of the patch. For example, a patch divided into 100 small patches, leading to 121 sets of (u, v) values, is shown in Fig. 5.3. We firstly pick one set of (u, v) as the initial value for one mesh node, if the aforementioned Newton iteration converges, then we continue to do projection for next mesh node. If the Newton method diverges, then we pick the next set of (u, v) from the grids in Fig. 5.3 as initial value, and run the Newton iterations again. This process is repeated until the parametric coordinates (u, v) for all the mesh nodes are found.

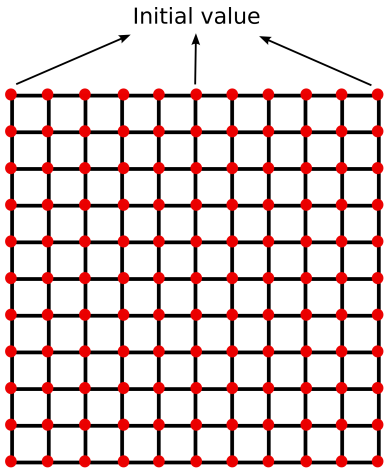


Figure 5.3: The choice of initial value in mesh projection.

The mesh projection algorithm in NSPCC is summarised in Algorithm 5.1:

Algorithm 5.1: Mesh projection

```
for Each mesh node do
    for Each moveable patch do
        Divide the patch into N × N small patches;
        Compute (N + 1) × (N + 1) sets of (u, v) values;
        for All (u, v) values do
            Choose one set of (u, v) as initial value;
            Perform the Newton iteration;
            if The point inversion algorithm is converged then
                Go to next mesh node;
            else
                Use the next set of (u, v) as initial value;
                Run the Newton iteration;
            end
        end
    end
end
```

One thing should be noted is that, although more grids used when setting the initial value give more accurate results, computational cost also needs to be considered, especially when there are many patches in the geometry and the surface mesh is fine. From the author's experience, there is no need to use very large number of initial value grids. $N = 20$ or 30 is more than enough for test cases used in this work. On the other hand, because the same parametric coordinates are used during the whole optimisation process, therefore the mesh projection will only have to be performance once. As

a result, its computational time is independent of the number of optimisation iterations, thus will not affect the whole run time of optimisation to a large extent. Indeed, as will be shown in Chapters 8 and 9, the run time of NSPCC in the whole optimisation is almost negligible compared to that of the flow solver and adjoint solver.

5.4 Continuity constraint

There are two kinds of continuity associated with parametric curves and surfaces, i.e. the parametric continuity and geometric continuity. For two curve segments, if they are joint together at end points, then the resulting curve is said to have G_0 continuity at the join. The resulting curve will have G_1 continuity if the tangent vector of both segments have share the same direction. Similar to G_0 continuity, if two curve segments are jointed together, the resulting curve then has C_0 continuity. For the parametric C_1 continuity, the tangent vector of two curve segments should have both the same direction and magnitude. The comparison between G_1 and C_1 continuity is illustrated in Fig. 5.4, where the difference can be seen clearly.

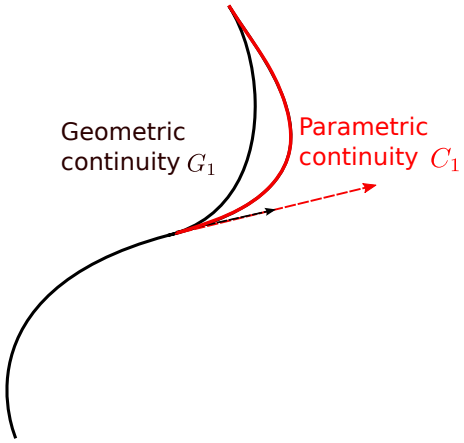


Figure 5.4: Continuity comparison between G_1 and C_1.

Obviously, parametric continuity is more restrictive than geometric continuity. For many applications, like those presented in this thesis, G_1 continuity is adequate. However, for those applications which depend on a smooth transition of reflected light, such as the car bodies, G_1 cannot satisfy the requirement. In these cases, at least C_2 should be achieved [52].

In this work, only geometric continuity is considered. However, the extension to parametric continuity is straightforward.

5.4.1 Geometric continuity

In previous section, the continuity has been introduced briefly using curves as example. In this work, we consider the geometric continuity among surfaces.

In shape modelling, normally several or a lot of surfaces are needed to describe a geometry. The interfaces between surfaces usually require at least G_0 continuity (adjacent surfaces touch or intersect to be watertight). In some applications such as CFD design, smoothness of the geometry is a very important aspect because the changes in tangency and curvature often have a very strong influence on the pressure field. Therefore, higher continuity are sometimes required, such as G_1 and G_2 which require two patches share the same tangent direction or continuous curvature along the common edge, respectively. Examples of different levels of geometric continuity are shown in Fig. 5.5.

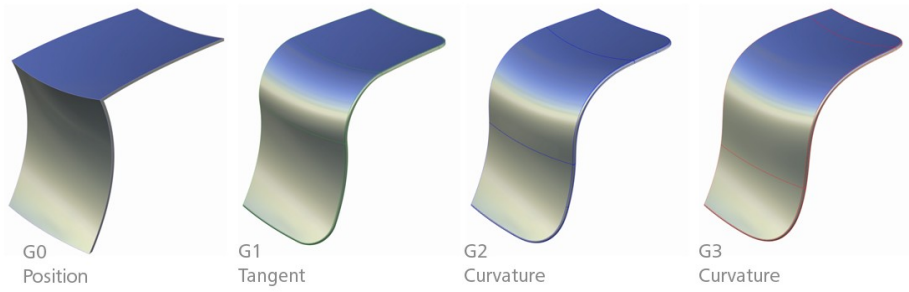


Figure 5.5: Different level of geometric continuity[3].

In the NSPCC approach, the geometric continuity between two fixed patches, or between fixed and free-to-move patches, are currently imposed by locking control points near the common edge. Specifically, to maintain G_0, G_1 and G_2 continuity, one, two and three rows control points are fixed, respectively. This is applied in Chapter 7 and Chapter 8.

The continuity constraint between two free-to-move patches is evaluated numerically at the same position in all touching or intersecting patches. Pairs of test points are evenly distributed along the common edges on both sides as illustrated in Fig. 5.6. The idea is that if the continuity constraints are satisfied on enough number of test points pairs, then they will be satisfied along the common edge [48].

[3]http://www.aliasworkbench.com/theoryBuilders/TB3_continuity1.htm

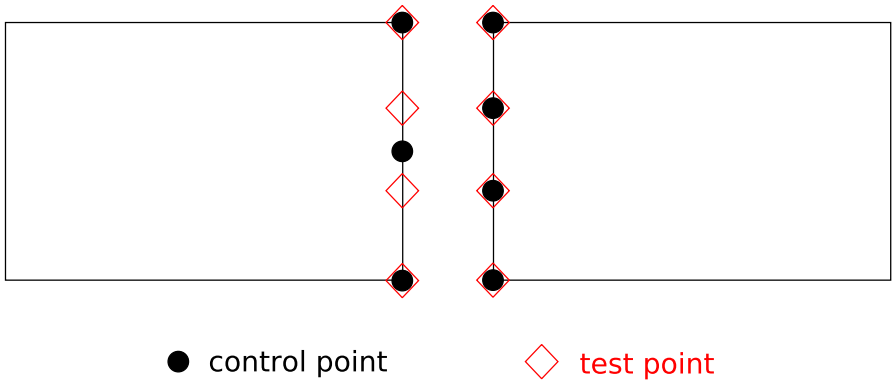


Figure 5.6: Two patches sharing one common edge.

Normally, the continuity can be expressed as

$$G_j = \xi_j^L - \xi_j^R = 0, \tag{5.8}$$

where j=0, 1, 2 corresponding to the G_0, G_1 and G_2 respectively. ξ_j^L and ξ_j^R are measurements on the left and right patch, such as the position, normal direction or curvature.

The G_0 continuity can then be written as

$$G_0 = X_{s,L} - X_{s,R} = 0, \tag{5.9}$$

where $X_{s,L}$ and $X_{s,R}$ are the test point on the left patch and right patch sharing an edge, respectively.

The G_1 and G_2 continuity are vector quantities, thus the constraints should be adjusted accordingly, and the normalised vectors can be used, so

$$G_j = \frac{\xi_j^L}{\|\xi_j^L\|} \pm \frac{\xi_j^R}{\|\xi_j^R\|} = 0. \tag{5.10}$$

where the sign, i.e. whether the vector should be added or subtracted is determined by how the patches are connected.

To be more specific, G_1 continuity can be written as

$$G_1 = \mathbf{n}_L \pm \mathbf{n}_R = \mathbf{0}, \tag{5.11}$$

where $\mathbf{n}_L$ and $\mathbf{n}_R$ are the unit normal vectors of the tangent plane at the test points on

either side of the patch interface. The unit normal vector is expressed as

$$\mathbf{n} = \frac{(\frac{\partial \mathbf{X}_s}{\partial u} \times \frac{\partial \mathbf{X}_s}{\partial v})}{\left\| \frac{\partial \mathbf{X}_s}{\partial u} \times \frac{\partial \mathbf{X}_s}{\partial v} \right\|}, \tag{5.12}$$

where u and v are the parametric coordinates of the surface as introduced in equation (2.1).

For G_1 continuity, a better way to express two parallel normal directions are

$$G_1 = \mathbf{n}_L \times \mathbf{n}_R = \mathbf{0}, \tag{5.13}$$

which has been implemented in NSPCC.

Let G denotes the continuity constraints of all required levels, and by linearising the constraints between two design iterations n and n+1 yields:

$$G^{n+1} \approx G^n + \sum_{i=1}^{N} \frac{\partial G}{\partial \mathbf{P}_i} \delta \mathbf{P}_i, \tag{5.14}$$

where $\delta \mathbf{P}_i$ is the displacement of the homogeneous coordinate of control point $\mathbf{P}_i$. To maintain the continuity constraints, let $G^{n+1} = G^n$, then

$$\sum_{i=1}^{N} \frac{\partial G}{\partial \mathbf{P}_i} \delta \mathbf{P}_i = \mathbf{C} \delta \mathbf{P} = 0. \tag{5.15}$$

The matrix $\mathbf{C}$ is called constraint matrix, and it has M_C rows and $4 \times N$ columns, where M_C is the number of constraint equations, N is the number of moveable NURBS control points. The AD tool Tapenade [150] is used in forward mode to obtain the entries in $\mathbf{C}$. Equation (5.15) indicates that the perturbation of the control points has to be in the nullspace (see Section 5.4.3) of the constraint matrix, to maintain the continuity constraints.

5.4.2 Imposing different continuity constraints to different edges

One important feature should have in the CAD-based parametrisation is to impose different geometric continuity to different edges. In this research, NSPCC is further developed such that different continuity level can be achieved in a single geometry. This allows NSPCC to be applied in more applications.

To achieve this, the constraint matrix is constructed as:

$$\mathbf{C} = \left[\begin{array}{cccc} \frac{\partial G_{0,1}}{\partial \mathbf{P}_1} & \frac{\partial G_{0,1}}{\partial \mathbf{P}_2} & \cdots & \frac{\partial G_{0,1}}{\partial \mathbf{P}_N} \\ \frac{\partial G_{0,2}}{\partial \mathbf{P}_1} & \frac{\partial G_{0,2}}{\partial \mathbf{P}_2} & \cdots & \frac{\partial G_{0,2}}{\partial \mathbf{P}_N} \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial G_{0,M_0}}{\partial \mathbf{P}_1} & \frac{\partial G_{0,M_0}}{\partial \mathbf{P}_2} & \cdots & \frac{\partial G_{0,M_0}}{\partial \mathbf{P}_N} \\ \hline \frac{\partial G_{1,1}}{\partial \mathbf{P}_1} & \frac{\partial G_{1,1}}{\partial \mathbf{P}_2} & \cdots & \frac{\partial G_{1,1}}{\partial \mathbf{P}_N} \\ \frac{\partial G_{1,2}}{\partial \mathbf{P}_1} & \frac{\partial G_{1,2}}{\partial \mathbf{P}_2} & \cdots & \frac{\partial G_{1,2}}{\partial \mathbf{P}_N} \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial G_{1,M_1}}{\partial \mathbf{P}_1} & \frac{\partial G_{1,M_1}}{\partial \mathbf{P}_2} & \cdots & \frac{\partial G_{1,M_1}}{\partial \mathbf{P}_N} \end{array}\right] = \left[\begin{array}{c} G_0 \\ \hline G_1 \end{array}\right] \tag{5.16}$$

where M_0 is the number of pairs of test points on edges imposed with G_0 continuity, M_1 is the number of pairs of test points on edges imposed with G_1 continuity. The subscript i of $G_{i,j}$ refers to G_i continuity constraints, subscript j denotes the j-th of M_0 pairs of test points for G_0 edges or M_1 pairs of test points for G_1 edges. As can be seen, the upper section of constraint matrix is about G_0 continuity, and the lower section is about G_1 continuity. In this way, different continuity can be imposed to different edges. Note that the G_1 edge should also satisfy G_0, therefore normally the value of M_0 is larger than that of M_1.

One thing should be noted about the constraint matrix is that, since the local control property of NURBS, only a small part of control points near the patches interfaces affect the continuity function value. Therefore, most elements in the constraint matrix are 0.

5.4.3 Introduction to nullspace and singular value decomposition

The shape optimisation is a constrained problem since geometric constraints are imposed. In the NSPCC approach, the optimisation problem is cast as an unconstrained one by using the projected gradient method [132, 217]. The basic idea of the projected gradient method is to restrict the search of design variables to the feasible subspace. In this work,

the nullspace of the constraint matrix $\mathbf{C}$ is utilised, as indicated by equation (5.15). To be more specific, to ensure the continuity constraints, we project the search direction to the nullspace of the constraint matrix $\mathbf{C}$. Therefore, the key to handle continuity constraint in the NSPCC approach is the *kernel* or *nullspace* of a matrix. The nullspace and its computation is briefly introduced here.

Let $\mathbf{B}$ denotes a matrix with dimension $m \times n$, $\mathbf{x}$ denotes a vector, then the nullspace of matrix $\mathbf{B}$ consists of all the vectors that satisfy the equation $\mathbf{Bx} = \mathbf{0}$. The nullspace of $\mathbf{B}$ is denoted as $N(\mathbf{B})$ or Ker($\mathbf{B}$). It is a subspace with dimension $n - r$, where r is the rank of $\mathbf{B}$.

One method to compute nullspace is the singular Value decomposition (SVD). The basic idea is that any $m \times n$ matrix $\mathbf{B}$ can be factored as:

$$\mathbf{B} = \mathbf{K}\mathbf{\Sigma}\mathbf{V}^T = (orthongonal)(diagonal)(orthogonal) \tag{5.17}$$

where $\mathbf{K}$ and $\mathbf{V}$ are unitary matrices, $\mathbf{\Sigma}$ a diagonal matrix. The dimension of $\mathbf{K}$ is m by m, its columns are eigenvectors of $\mathbf{B}\mathbf{B}^T$. The dimension of $\mathbf{V}$ is n by n, its columns are the eigenvectors of $\mathbf{B}^T\mathbf{B}$. The r singular values on the diagonal of $\mathbf{\Sigma}$ ($m \times n$) are the square roots of the non-zero eigenvectors of both $\mathbf{B}\mathbf{B}^T$ and $\mathbf{B}^T\mathbf{B}$. Here, the last $(n - r)$ columns of $\mathbf{V}$ give orthonormal bases for the nullspace of $\mathbf{B}$.

SVD is a very important technique in linear algebra and has been widely applied in many areas, such as image processing [218, 219], signal processing [220, 221], robotics [222], geometry parametrisation [71, 78], the least squares [223] and many others. There are several aims to use SVD in this work. Firstly, although in theory the rank of the matrix is not difficult to determine, in numerical computations it is non-trivial due to round-off errors [224]. SVD is a most-widely used method for numerical rank determination [225], therefore it is utilised. Secondly, it is used to obtain the nullspace of constraint matrix. The columns of this nullspace are actually shape deformation modes (see Section 5.4.5). Finally, SVD is also used to compute the pseudo inverse of a matrix in the continuity recovery steps (see Section 5.5).

More details about nullspace and SVD can refer to [226].

5.4.4 Computing nullspace of the constraint matrix

The equation (5.15) indicates that the displacement of the control points have to lie in the nullspace of the constraint matrix $\mathbf{C}$ to maintain the geometric continuity. In this work, the nullspace is computed using the standard SVD algorithm from the LAPACK

library [227–229].

As mentioned in Section 5.4.1, the dimension of the constraint matrix $\mathbf{C}$ is $M_C \times 4N$, where M_C is the number of constraint equations, and N is the number of control points, then the SVD of $\mathbf{C}$ is

$$\mathbf{C} = \mathbf{K}\Sigma\mathbf{V}^T, \tag{5.18}$$

where $\mathbf{K}$ is a $M_C \times M_C$ unitary matrix, Σ is a $M_C \times 4N$ diagonal matrix with non-negative real numbers on the diagonal, and the $4N \times 4N$ unitary matrix $\mathbf{V}^T$ denotes the transpose of $\mathbf{V}$. In theory, the last $(4N - r)$ columns of the matrix $\mathbf{V}$ span the nullspace of $\mathbf{C}$ [230], here r is the theoretical rank of the matrix, i.e. the number of non-zero diagonal entries in Σ. However, in numerical computations it is non-trivial to determine the rank in the presence of round off errors [224], and real arithmetic can be misleading [230]. Due to finite-precision arithmetic, the singular values do not drop off to zero sharply after all non-singular modes, but show a rather gradual decrease. Therefore, a cut-off threshold value σ_C is used to determine the rank used in practice, termed numerical rank r'. The nullspace corresponding to this numerical rank is termed numerical nullspace, denoted by $\mathrm{Ker}(\mathbf{C})$ [225].

The perturbation of control points can be computed as a linear combination of columns of numerical nullspace, to ensure the perturbation lies in the numerical nullspace, i.e.

$$\delta\mathbf{P} = \sum_{k=1}^{4N-r'} v_{k+r'}\delta\alpha_k = \mathrm{Ker}(\mathbf{C})\delta\boldsymbol{\alpha}, \tag{5.19}$$

where $\delta\alpha_k, k = 1, 2, ..., 4N - r'$ are the perturbations of design parameters, $v_1, v_2, ...,$ $v_{k+r'}$ are the columns of the numerical nullspace. The perturbation of the control points can also be written as

$$\delta\mathbf{P} = \mathrm{Ker}(\mathbf{C})\delta\boldsymbol{\alpha} = \sum_{i=r'+1}^{r} v_i\delta\alpha_i + \sum_{i=r+1}^{4N} v_i\delta\alpha_i, \tag{5.20}$$

where r is the theoretical rank and r' is the numerical rank as mentioned earlier. The first term on the right hand side is corresponding to those singular values which are below the cut-off value while larger than 0. This term leads to the inaccuracy of the nullspace. The second term is in the exact theoretical nullspace. Theoretically, only the 2nd term should be used to maintain the continuity constraints, however, it is difficult to determine the value of r in numerical computing. In addition, to have a larger design space, we usually lump these two terms together and regard them together as the numerical nullspace.

The columns corresponding to the first term and second term in equation (5.20) are

shown in red and green in the following matrix $\mathbf{V}$, respectively:

$$\mathbf{V} = \underbrace{\underbrace{\begin{bmatrix} v_{11} & \cdots & v_{1(r'+1)} & \cdots & v_{1r} & v_{1(r+1)} & \cdots & v_{1(4N)} \\ v_{21} & \cdots & v_{2(r'+1)} & \cdots & v_{2r} & v_{2(r+1)} & \cdots & v_{2(4N)} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ v_{(4N)1} & \cdots & v_{(4N)(r'+1)} & \cdots & v_{(4N)r} & v_{(4N)(r+1)} & \cdots & v_{(4N)(4N)} \end{bmatrix}}_{\text{numerical nullspace}}}_{\text{exact nullspace}}$$

Written in matrix form, equation (5.20) becomes

$$\delta\mathbf{P} = \begin{bmatrix} v_{1(r'+1)} & \cdots & v_{1r} & v_{1(r+1)} & \cdots & v_{1(4N)} \\ v_{2(r'+1)} & \cdots & v_{2r} & v_{2(r+1)} & \cdots & v_{2(4N)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ v_{(4N)(r'+1)} & \cdots & v_{(4N)r} & v_{(4N)(r+1)} & \cdots & v_{(4N)(4N)} \end{bmatrix} \begin{bmatrix} \alpha_{r'+1} \\ \vdots \\ \alpha_r \\ \alpha_{r+1} \\ \vdots \\ \alpha_{4N-r'} \end{bmatrix}. \tag{5.21}$$

After computing $\delta\mathbf{P}$, the new NURBS surface is

$$\mathbf{S}_{new} = \mathbf{S}_{initial} + \Delta\mathbf{S}, \tag{5.22}$$

where $\mathbf{S}_{initial}$ is the initial surface, $\mathbf{S}_{new}$ is the updated surface and $\Delta\mathbf{S}$ is computed by

$$\Delta\mathbf{S} = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\delta\mathbf{P}_{i,j}. \tag{5.23}$$

One thing should note is that, this is the NURBS surface in homogeneous space (the superscript is omitted for simplicity), thus a projection step is needed to obtain NURBS surface in 3-D space.

5.4.5 Shape deformation modes in the nullspace

The equation (5.19) indicates that the displacement of control points is a linear combination of the columns of the numerical nullspace. Actually, each column in the numerical nullspace is corresponding to one deformation mode, which should satisfy the continuity constraints. Therefore, the real design space in the NSPCC approach is the liner combination of these deformation modes. Note that the matrix $\mathbf{V}^T$ in equation (5.18) is a unitary matrix, which means the columns in $\mathbf{V}^T$ are orthogonal to each other. As

a consequence, the deformation modes in the nullspace are orthogonal to each other as well.

Also note that in Section 2.3.1.2, some works were reviewed which use SVD to extract deformation modes. In those studies, the main aim of using SVD is to extract determinant deformation modes from a training library consists of hundreds or thousands of airfoils, such that the dimension of design space can be reduced. In this thesis, the aim of using SVD is different as stated in Section 5.4.3. In addition, in the NSPCC approach, no training library is needed.

The half-cylinder shown in Fig. 1.5 consists of two NURBS surfaces and 20 control points. For convenience, it is shown here again in Fig. 5.7. G_0 continuity is imposed along the common edge. The deformation modes corresponding to first two columns in the numerical nullspace are shown in Fig. 5.8. The figure clearly indicates that different columns result in different perturbation of control points.

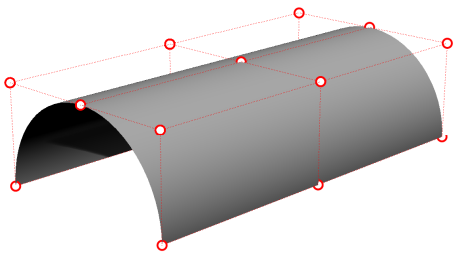


Figure 5.7: A half-cylinder geometry and its control points.

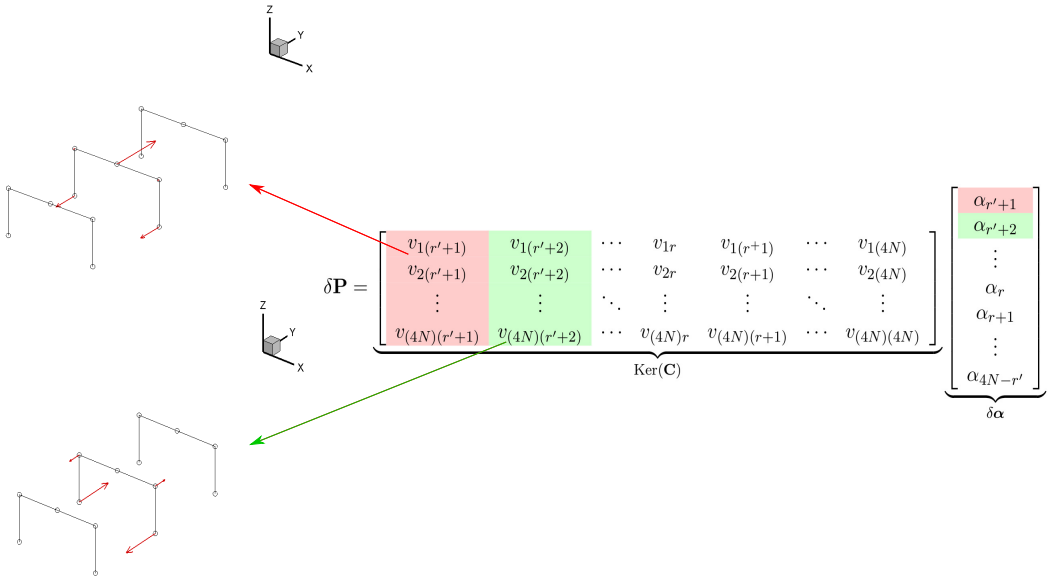


Figure 5.8: Deformation modes corresponding to the columns in nullspace of the constraint matrix.

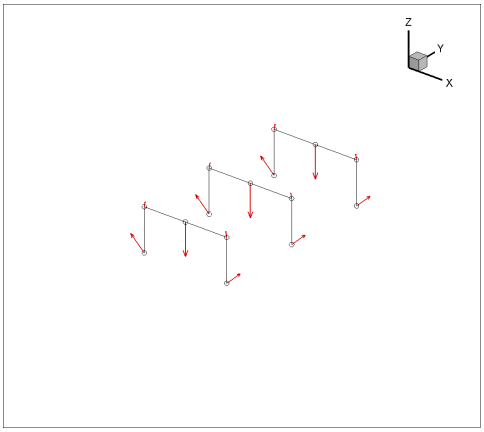
More deformation modes in the nullspace are illustrated in Fig. 5.10. It can be seen that in all these deformation modes, the G_0 continuity is satisfied. The displacement of the control points is then the linear combination of these deformation modes, i.e.

$$\delta\mathbf{P} = \delta\alpha_1 \cdot \text{mode 1} + \delta\alpha_2 \cdot \text{mode 2} + \cdots + \delta\alpha_i \cdot \text{mode i} + \cdots + \delta\alpha_{4N-r\prime} \cdot \text{mode } (4\text{N} - r\prime). \quad (5.24)$$

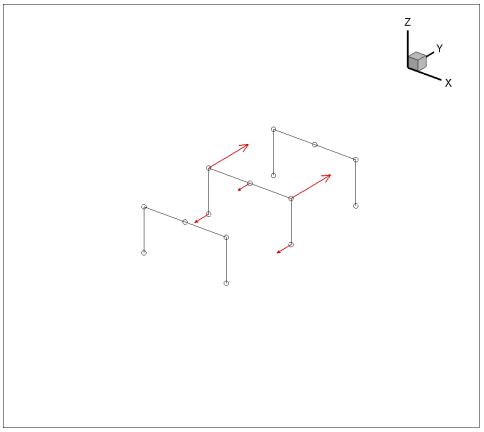
Therefore, the new position of the control points are:

$$\mathbf{P}^{new} = \mathbf{P}^{old} + \sum_{i=1}^{4N-r\prime} \delta\alpha_i \text{mode i}. \quad (5.25)$$

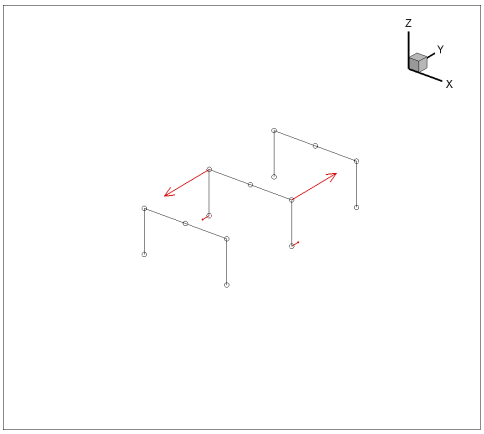
These linear combination coefficients are actually used as design variables, termed *mathematically-derived design variables*. In the optimisation framework, these coefficients are optimised, then we can obtain the perturbation of control points from equation (5.19). It can be seen that the idea of using a linear combination of various basis of NSPCC is similar to some approaches introduced in Chapter 2, such as the PARSEC method and SVD method presented in Section 2.3.1.2. The difference lies in the basis functions (deformation modes). In addition, NSPCC can be applied to 3-D geometries.



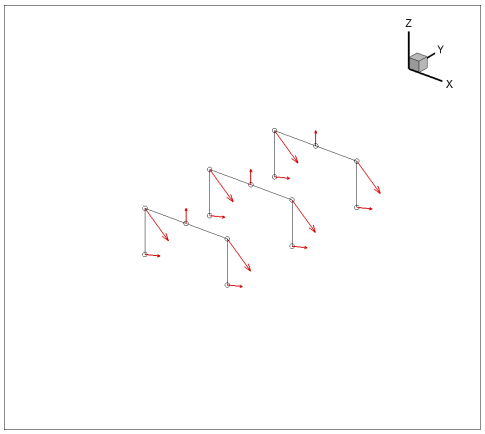
(a) Mode 4



(b) Mode 5



(c) Mode 6



(d) Mode 7

Figure 5.10: Some deformation modes in the numerical nullspace of the half-cylinder geometry.

More discussion on the deformation mode will be presented in the S-bend air duct optimisation case in Chapter 7.

5.4.6 Computation of CAD sensitivity

Based on equation (5.19), the gradient of the surface points w.r.t. design variables considering geometric continuity constraints, i.e. the CAD sensitivity, can be written as

$$\frac{\partial \mathbf{X}_s}{\partial \boldsymbol{\alpha}} = \frac{\partial \mathbf{X}_s}{\partial \mathbf{P}} \frac{\partial \mathbf{P}}{\delta \boldsymbol{\alpha}} = \frac{\partial \mathbf{X}_s}{\partial \mathbf{P}} \mathrm{Ker}(\mathbf{C}). \tag{5.26}$$

The term $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}}$ is computed by using AD in forward mode, because normally there are much more surface points than control points. As mentioned in Section 5.4.1, the entries in the constraint matrix $\mathbf{C}$ are computed also using forward mode AD. Therefore, in the current implementation of NSPCC approach, the CAD sensitivity is obtained through AD in forward mode.

The gradient of the cost function w.r.t. design variables is then

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{dJ}{dX_s}\frac{\partial X_s}{\partial \mathbf{P}}\frac{\partial \mathbf{P}}{\delta \boldsymbol{\alpha}} = \frac{dJ}{d\mathbf{X}_s}\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}}\mathrm{Ker}(\mathbf{C}). \tag{5.27}$$

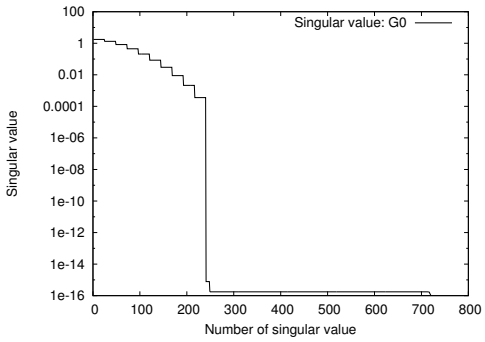
where $\frac{dJ}{d\mathbf{X}_s}$ comes from the adjoint solver introduced in Section 4.3.1.

In node-based parametrisation approach, the sensitivities of the objection function w.r.t. surface nodes, $\frac{dJ}{d\mathbf{X}_s}$, are used. In CAD-based method, the gradient is extended to control point $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}}$, this is what other works do [50, 105]. To handle the continuity between patches, again the gradient is extended to include the numerical nullspace $\mathrm{Ker}(\mathbf{C})$.

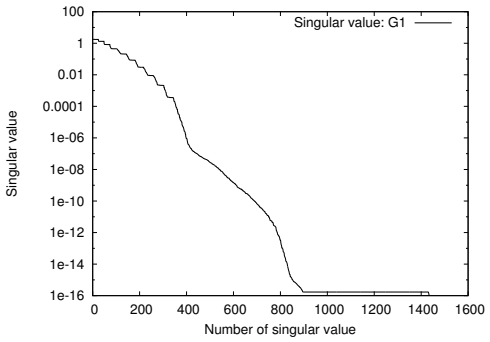
5.4.7 The size of design space

The equation (5.19) indicates the number of design parameters in $\boldsymbol{\alpha}$ is $(4N - r')$, where N is the number of moveable control points on design surfaces and r' is the numerical rank. For a specific case, the number of control points N is a constant (refinement during the optimisation is not considered), so the number of design parameters will be dependent on r'. Therefore, the design space will be affected by the value of r'.

In this work, it is found that there is a clear gap in the distribution of singular value of the constraint matrix $\mathbf{C}$, if the continuity constraints are linear, such as the G_0 continuity when weights are fixed as illustrated in Fig. 5.11(a). In this case, the numerical rank is easy to determine, and normally $r = r'$. However, if the constraints are non-linear, such as the G_1 continuity, there is no such a clear jump in the singular values, instead they will decrease to 0 gradually as displayed in Fig. 5.11(b). As a consequence, the aforementioned cut-off threshold value σ_C is needed to determine r'. The value of numerical rank r' will then be dependent on σ_C. In this case, if smaller σ_C is used, r' will be larger and closer to the theoretical rank r, so the continuity constraints will be satisfied more restrict. However, the design space will then become smaller because fewer deformation modes are contained in the design space. On the other hand, if larger σ_C is used, r' will be smaller and away from r. Even though the continuity constraints will then be violated slightly because of the inaccuracy of the nullspace, more deformation modes are involved hence the design space is larger. Therefore, the choice of σ_C is actually a



(a) The singular value distribution when imposing only G_0 continuity constraint.



(b) The singular value distribution when imposing G_1 continuity constraint.

Figure 5.11: Singular values when imposing different continuity constraints.

trade-off between the continuity constraints and design space.

Assuming σ_C changes and the numerical rank becomes $r'+1$, the nullspace will then be the last $(4N-r'-1)$ columns of $\mathbf{V}$ and one deformation mode is removed from the design space, thus $\delta\mathbf{P}$ is

$$\delta\mathbf{P} = \begin{bmatrix} v_{1(r'+2)} & \cdots & v_{1r} & v_{1(r)} & \cdots & v_{1(4N)} \\ v_{2(r'+2)} & \cdots & v_{2r} & v_{2(r)} & \cdots & v_{2(4N)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ v_{(4N)(r'+2)} & \cdots & v_{(4N)r} & v_{(4N)(r)} & \cdots & v_{(4N)(4N)} \end{bmatrix} \begin{bmatrix} \alpha_{r'+2} \\ \vdots \\ \alpha_r \\ \alpha_{r+1} \\ \vdots \\ \alpha_{4N-r'} \end{bmatrix}. \tag{5.28}$$

Results illustrating the effect of σ_C on optimisation performance will be presented in Chapter 7.

The number of design parameters affect the choice of optimiser. Indeed, in the NSPCC approach, if geometric continuity constraints are not imposed or they are not imposed with the test point-based approach, the last term in equation (5.27) disappears. In this case, the control points are directly used as design variables, instead of the linear combinational coefficients. Since the number of control points is a constant (refinement inside optimisation is not considered), thus the size of the design vector is a constant.

However, if the constraints are imposed through test points and the constraint matrix is built, then the number of design variables (the linear combinational coefficients) may not be a constant, especially when then constraints are non-linear. This is because there is not clear gap in the singular value as mentioned above, thus when the constraint matrix is updated in each optimisation iteration, the numerical rank r' is possible to

change. As a result, the dimension of the numerical nullspace, thus the number of columns (deformation modes) may change. The illustration of this change will be presented in the U-bend optimisation case in Chapter 8.

As a result of the changing number of design variables, previously only the steepest descent method which does not require the history gradient information can be used as optimiser [48, 49, 167]. Advanced optimisers, such as the Newton and quasi-Newton methods, do not work if no measures are taken.

In this study, a method to apply BFGS algorithm as optimiser with NSPCC is proposed and investigated. Generally speaking, the idea is freezing the parametrisation for a number of design iterations, as illustrated in Fig. 5.12. In other words, the term $\frac{\partial \mathbf{X}_s}{\partial \boldsymbol{\alpha}}$ in equation (5.26) is frozen within these iterations. As a consequence, during these iterations, the number and content of the deformation modes are frozen, which in turn allows using the BFGS algorithm that builds up an approximation to the inverse Hessian matrix. Note that the continuity recovery steps presented in Section 5.5 are still needed when the design space is fixed, such that the continuity constraints will not be violated. The other thing is about the choice of fixed steps n. If n is too small the BFGS may not be able to approximate the Hessian matrix well, while if n is too large the geometric sensitivities are not updated in time such that the final results will be affected. From the author's experience, $n = 4$ or $n = 5$ should be a reasonable choice. This idea is applied to the optimisation of a U-bend cooling channel, and the results will be presented in Chapter 8.

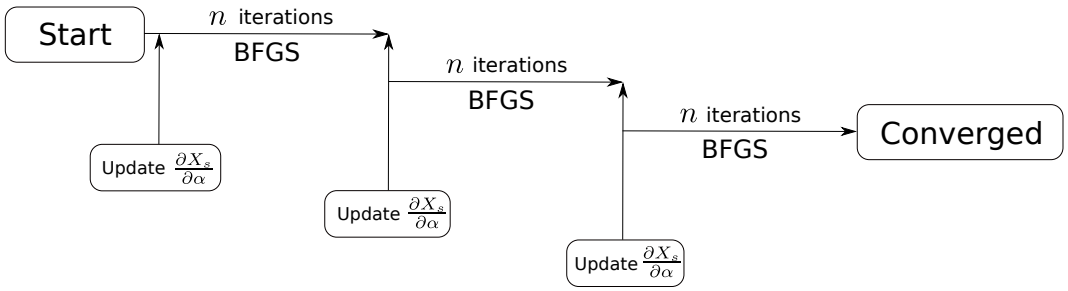


Figure 5.12: Applying the BFGS algorithm with NSPCC.

5.4.8 The effective rank

Based on above discussion in Section 5.4.7, one can see that it is important to compute a proper numerical rank which can offer a relative large design space while does not violate the continuity constraints too much. Otherwise, the continuity recovery step (see Section 5.5) will not be able to recover the continuity.

From the mathematical point of view, rank is the number of independent rows or columns. However, this could be hard to decide in real arithmetic due to the presence of round-off errors. Previously, trial and errors are needed to determine a cut-off value thus the rank which can give good performance. This is very tedious and time-consuming, because only in the optimisation loop can one find out whether a cut-off value σ_C is proper or not, therefore the expensive flow and adjoint solver are involved. Besides, the choice of σ_C is actually case-dependent. As a result, the trial and errors are needed for each new case.

We propose to determine the numerical rank from $\mathbf{C}\mathbf{C}^T$ or $\mathbf{C}^T\mathbf{C}$ instead of the constraint matrix $\mathbf{C}$. Theoretically, both $\mathbf{C}\mathbf{C}^T$ and $\mathbf{C}^T\mathbf{C}$ share the same rank with $\mathbf{C}$, but in computations they provide a more stable and effective rank [230]. We found that this effective rank can provide a reasonable and automatic pre-conditioner in computing the numerical nullspace. By using the effective rank, a large design space is obtained, while at the same time non-linear continuity constraints can still be recovered.

The effective rank is determined as per following procedures in this study:

1. Compute $\mathbf{C}\mathbf{C}^T$ or $\mathbf{C}^T\mathbf{C}$, and denote as $\mathbf{C}^*$.

2. Calculate the singular values of $\mathbf{C}^*$ using LAPACK, and denote as $svd(\mathbf{C}^*)$.

3. Compute the cut-off value following the default algorithm utilised in MATLAB[4]:

$$tol = max(size(\mathbf{C}^*)) * svd(1) * eps, \tag{5.29}$$

   where $svd(1)$ is the largest singular value of $\mathbf{C}^*$, eps is the machine precision. The value of eps is approximately $2.2204 * 10^{-16}$ for double precision and $1.1921 * 10^{-7}$ for single precision on machines that support IEEE floating point arithmetic[5]. Throughout this research, double precision is used.

4. The effective rank is then the number of singular values in $svd(\mathbf{C}^*)$ which are larger than tol.

Figure 5.13 presents a comparison between the singular values of $\mathbf{C}\mathbf{C}^T$ and $\mathbf{C}$ for an S-bend air duct optimisation case (see Chapter 7). It indicates clearly that the singular values of $\mathbf{C}\mathbf{C}^T$ decrease to be smaller than 10^{-14} very quickly compared to those of $\mathbf{C}$. The steeper the curve is, the easier it is to decide rank. This demonstrates that a more stable rank, namely the proposed effective rank, could be obtained from the singular value of $\mathbf{C}\mathbf{C}^T$ or $\mathbf{C}^T\mathbf{C}$. What's more, the tolerance defined in equation (5.29)

[4]https://uk.mathworks.com/products/matlab.html
[5]https://www.gnu.org/software/octave/doc/v4.0.1/Mathematical-Constants.html

has taken the case information, namely the size of constraint matrix and the maximum singular value, into consideration. Therefore, the effective rank can adapt to new cases automatically. The feasibility of this effective rank will be demonstrated in practical optimisation case presented in Chapter 7 and Chapter 8.

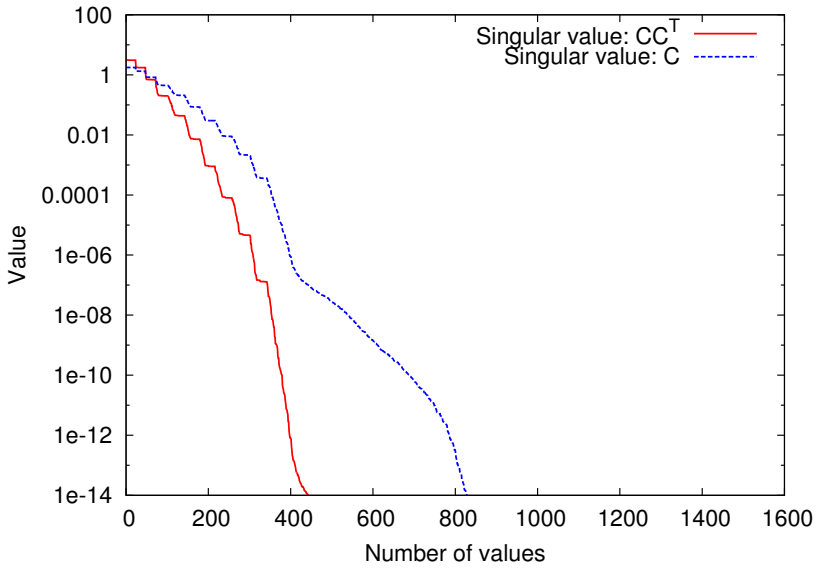


Figure 5.13: The comparison of singular values of $\mathbf{CC}^T$ and $\mathbf{C}$.

5.4.9 Required number of test points

The continuity is maintained through enough number of test points in the NSPCC approach. Therefore, the number of test points is a key point. Xu et al. [48] suggested running a series of SVD computations with different number of test points, until the number of non-zero singular value doesn't change any further. This is time consuming and labour-intensive. Therefore, we investigate the factors that have influence on the number of test points and propose a method to estimate the required number. This is helpful to understand the test point-based approach and save time for setting up.

Because the B-splines are polynomial so a section of B-spline curve of degree N_q on the patch edge, can be matched uniquely if there are N_q+1 distinct test points within the knot vector interval that supports this section of the curve. Therefore, the number of required test point pairs for a B-spline curve can be determined a-priori by considering each non-zero knot-interval.

The relationship between the number of knots N_k, the number of control points N_p

and the degree of the spline N_q is

$$N_k = N_p + N_q + 1. \tag{5.30}$$

The number of non-zero knot intervals can be obtained as

$$N_i = (N_k - 1) - 2N_q - N_M \tag{5.31}$$

where N_M is the number of zero knot intervals because of internal multiplicities. From equation (5.30) and (5.31), the number of non-zero knot intervals becomes

$$N_i = N_p - N_q - N_M. \tag{5.32}$$

In each interval we then need N_q+1 test points to fit the polynomial exactly, for the left side of a patch edge with N_p control points, we hence need M_L test points

$$M_L \geq (N_q+1)(N_p - N_q - N_M).$$

The same relation is valid for the right side M_R. Assuming a regular spacing of knots, the number of required test points $M_{T,E}$ along edge E then becomes

$$M_{T,E} \geq \max(M_L,\, M_R). \tag{5.33}$$

To allow for non-regular knot-intervals we use

$$M_{T,E} \geq f_T \max(M_L,\, M_R)$$

with an inflation factor f_T chosen around $1.0 \leq f_T \leq 1.5$ according to the author's practical experience. In the typical case of equal polynomial orders $N_q|_L = N_q|_R$ this becomes

$$M_{T,E} \geq f_T(N_q+1)(\max(N_p|_L,\, N_p|_R) - N_q - N_M). \tag{5.34}$$

The exactness argument employed for B-splines does not carry over straightforwardly to the non-rational functions in NURBS. This could be accommodated by increasing the factor f_T.

In addition, it is found that the continuity level imposed affects the required number of test points. If only G_0 continuity is imposed, then the above formulation is fine. However, more test points are needed if higher level continuity is required.

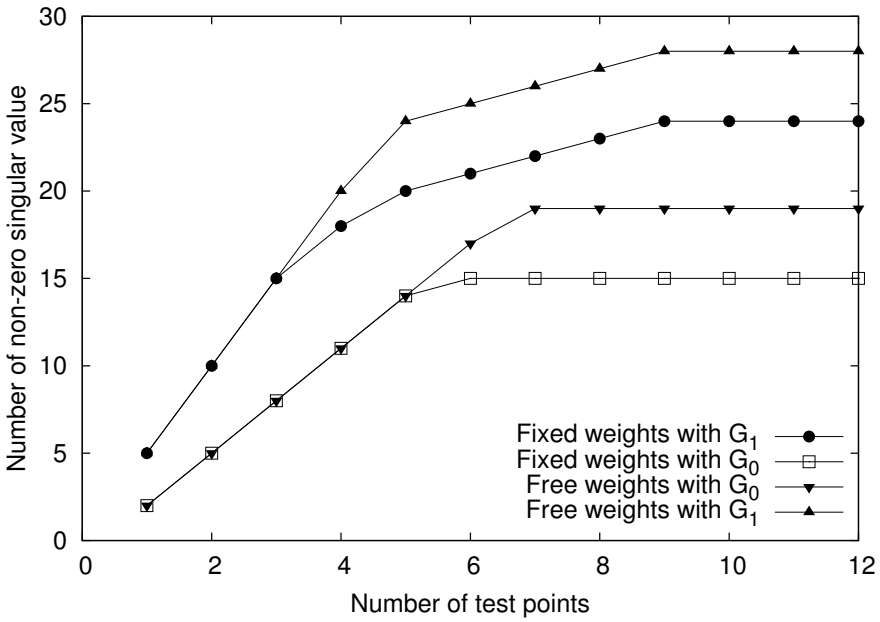


Figure 5.14: The number of non-zero singular value when the number of test points changes for half-cylinder case.

Along the common edge of the aforementioned half-cylinder in Fig. 5.7, the number of control points on each surface is 5, the degree is 2. The knot vector is

$$\{0, 0, 0, 0.5, 0.5, 1.0, 1.0, 1.0\},$$

hence the number of non-zero knot vector interval is 2, matching the estimate of $N_p - N_q - N_M = 5 - 2 - 1 = 2$ in equation (5.32). According to equation (5.34), the number of test points when B-spline surfaces used and G_0 imposed should be

$$M \geq f_T(2 + 1)2 = 6f_T.$$

In this case, the knot vector on both sides are the same and regularly spaced, therefore $f_T = 1$ can be used if the weights of control points are fixed. Figure 5.14 shows the relationship between the number of non-zero singular values and test points. It can be observed that the number of non-singular values no longer increases beyond 6 test points, if the weights of control points are fixed and only G_0 continuity is imposed. This means that 6 test points are enough to ensure the G_0 continuity, which matches very well with the result of equation (5.34). It can also be seen that more test points are required before the number of non-zero singular value stops to increase, if the weights are free or higher continuity is imposed. This has good agreement with the previous discussion.

5.5 Continuity recovery step

In the NSPCC approach, continuity recovery steps are required to ensure the continuity constraints are satisfied, because the continuity may be violated for two reasons. Firstly, because of round-off errors, the numerical nullspace is used in NSPCC instead of the exact theoretical nullspace. The inaccuracy of nullspace, i.e. the first term on the right hand side of equation (5.20), will lead to slight violation of the continuity constraints. In Fig. 5.15, the theoretical nullspace is depicted by the solid line with arrow, while the numerical nullspace is illustrated by the dashed line with arrow. Secondly, the G_1 or higher geometric continuity constraint are non-linear, but the projected gradient method in the nullspace is only a first-order approximation, thus the perturbation in nullspace will violate the geometric continuity. These two factors of violation are labelled as area 1 and area 2 in Fig. 5.15, respectively.

Since any vector can be split in to a row space component and a nullspace component which are perpendicular to each other [230], and the nullspace component is used in NSPCC, thus the recovery steps will be in the row space of the constraint matrix $\mathbf{C}$.

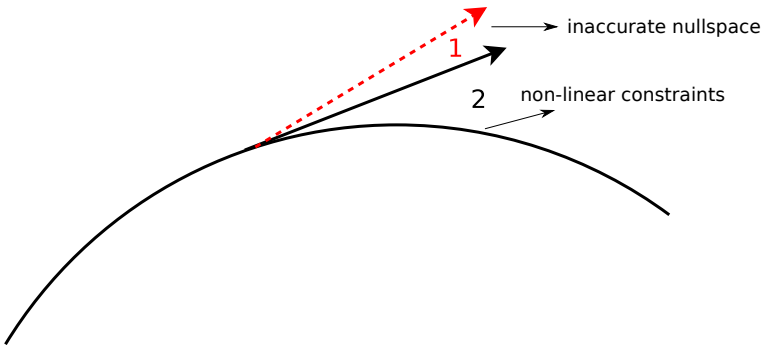


Figure 5.15: Two factors that violate the geometric continuity constraints.

Assuming $\delta\mathbf{P}_r$ is the recovery displacement of control points, δG_1 is the deviation of the G_1 constraint function from the target due to both the inaccuracy of nullspace and non-linear property of G_1 continuity. Then $\delta\mathbf{P}_r$ should eliminate δG_1 while maintaining G_0. Therefore, we have

$$\delta\mathbf{P}_r = \mathrm{Ker}(\mathbf{C}_0)\delta\alpha_r, \tag{5.35}$$

and

$$\mathbf{C}_1\delta\mathbf{P}_r + \delta G_1 = 0, \tag{5.36}$$

where $\mathbf{C}_0$ is the matrix which only contains entries related to G_0 continuity and $\mathrm{Ker}(\mathbf{C}_0)$ is its nullspace. $\mathbf{C}_1$ is the matrix which only contains entries related to G_1 continuity.

Substituting equation (5.35) into (5.36) yields

$$\mathbf{C}_1\mathrm{Ker}(\mathbf{C}_0)\delta\alpha_r + \delta G_1 = 0. \tag{5.37}$$

We firstly solve equation (5.37) to obtain $\delta\alpha_r$, and then substitute $\delta\alpha_r$ into equation (5.35) to compute $\delta\mathbf{P}_r$. $\delta\mathbf{P}_r$ will recover the deviation of G_1 with G_0 satisfied.

In equation (5.37), there are $3M_1$ equations and $(4N - r'')$ unknowns, where M_1 is the number of test points imposed on G_1 edges, N is the number of moveable control points on the design surfaces, r'' is the numerical rank of $\mathbf{C}_0$. The relationship between be number of equations and unknowns may change if the value of $3M_1$ and $(4N-r'')$ change, therefore it is hard to say whether this equation system has exact solution or not. But from the authors' practical experience, quite frequently there are more unknowns than equations. For example, in the S-bend case which will be presented in Chapter 7, there are 720 equations and 1360 unknowns in equation (5.37). Therefore, equation (5.37) has dependent columns, so the minimum length solution, which is also the optimal solution of equation (5.37), should be found [230]. The pseudo inverse will thus be used.

The pseudo inverse of $\mathbf{C}_1\mathrm{Ker}(\mathbf{C}_0)$ is obtained by using SVD. Then we have

$$\delta\alpha_r = -\left[\mathbf{C}_1\mathrm{Ker}(\mathbf{C}_0)\right]^{+}\delta G_1. \tag{5.38}$$

Thus

$$\delta\mathbf{P}_r = -\mathrm{Ker}(\mathbf{C}_0)\left[\mathbf{C}_1\mathrm{Ker}(\mathbf{C}_0)\right]^{+}\delta G_1. \tag{5.39}$$

One thing should be noted is that in previous work where only B-splines is used [48], the matrix $\mathbf{C}_0$ and $\mathrm{Ker}(\mathbf{C}_0)$ are independent of $\delta\alpha_r$, thus only need to computed and stored at the beginning of the optimisation. However, when the homogeneous form of NURBS is used and the weights are free to change, $\mathbf{C}_0$ and $\mathrm{Ker}(\mathbf{C}_0)$ will no longer be independent of $\delta\alpha_r$, thus need to be updated in each step. Besides, $\mathbf{C}_1$ and δG_1 are dependent on $\delta\alpha_r$ as well. Therefore, equation (5.37) is a non-linear system, thus iterative method is required to solve it. The number of recovery steps needed are determined by the deviation value from exact G_1 continuity. The recovery step will stop if the deviation value from exact G_1 is taken back to below the chosen threshold value.

The continuity recovery algorithm is summarised in Algorithm 5.2. Simple geometric cases demonstrating the effectiveness of recovery step will be given in Section 6.4.1, and the result of G_1 recovery in practical shape optimisation case will be shown in Chapter

7 and Chapter 8.

Algorithm 5.2: Continuity recovery step

```
while G1 deviation is larger than threshold do
    read C1, KerC0, δG1;
    calculate C1KerC0 ;
    calculate δαr ;
    calculate δPr ;
    add δPr to δP ;
    if Weights are fixed then
        update C1, δG1;
    else
        update C1, KerC0, δG1;
    end
    read G1 deviation value;
end
```

5.6 Scaling of control points

The scale and effect of the design variables will have a significant influence on the convergence toward the optimum. In optimisation, a problem is said to be poorly scaled if changes to design variables in one direction produce much larger variations in the value of cost function, than do changes to design variables in another direction [131]. Considering e.g. a curve aligned with the x-axis. Control point movements in the curve-normal y direction will have a much stronger effect on the shape than movements in the tangential x-direction. Similarly, the weights have scales around unity, while the scaling of the coordinates entirely depends on the measurement units used for the coordinate values.

In practice, if different kinds of variables with different scales are used in the design vector, scaling is necessary. For example, use the control point position and the angle of attack as design variables at the same time [110, 231, 232]. Poorly scaling will be harmful for the performance of optimiser.

In this work, the so called diagonal scaling [131] is implemented and will be introduced here. Another scaling method is implemented for the ONERA M6 case and will be introduced in Chapter 9.

For a particular control point $\mathbf{P}_i$, the scaled control point is obtained from

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \\ \Omega_i \end{bmatrix} = \begin{bmatrix} \frac{1}{w_i^* X_{MAX}^*} & 0 & 0 & 0 \\ 0 & \frac{1}{w_i^* Y_{MAX}^*} & 0 & 0 \\ 0 & 0 & \frac{1}{w_i^* Z_{MAX}^*} & 0 \\ 0 & 0 & 0 & \frac{1}{w_i^*} \end{bmatrix} \begin{bmatrix} \omega_i x_i \\ \omega_i y_i \\ \omega_i z_i \\ \omega_i \end{bmatrix} \tag{5.40}$$

where $(X_i, Y_i, Z_i, \Omega_i)$ is the scaled control point, ω_i^* is the initial weight, X_{MAX}^*, Y_{MAX}^*, Z_{MAX}^* are the maximum length of design surfaces in the (x, y, z) direction, respectively. Note that the value of $X_{MAX}^*, Y_{MAX}^*, Z_{MAX}^*$ can be changed to implement different scaling performances.

Then for all control points, we have

$$\mathbf{P}^s = \mathbf{L}\mathbf{P}^\omega \tag{5.41}$$

where $\mathbf{L}$ is a diagonal scaling matrix consists of many small matrices with the form in equation (5.40), $\mathbf{P}^s$ contains the scaled control points, $\mathbf{P}^\omega$ contains the original control points in homogeneous form. As a consequence, the displacement of the homogeneous control points need to be obtained by

$$\delta\mathbf{P}^\omega = \mathbf{L}^{-1}\delta\mathbf{P}^s. \tag{5.42}$$

where $\delta\mathbf{P}^s$ is the perturbation of scaled control points, given by the optimisation loop.

The derivative of cost function, equation (5.27), now changes to

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \mathbf{X}_s}\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^s}\frac{\partial \mathbf{P}^s}{\partial \boldsymbol{\alpha}}, \tag{5.43}$$

where the term $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^s}$ needs to be computed according to equation (5.42) as

$$\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^s} = \frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^\omega}\frac{\partial \mathbf{P}^\omega}{\partial \mathbf{P}^s} = \frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^\omega}\mathbf{L}^{-1}. \tag{5.44}$$

In addition, the elements in the constraint matrix $\mathbf{C}$ also need to be modified using

$$\frac{\partial G}{\partial \mathbf{P}^s} = \frac{\partial G}{\partial \mathbf{P}^\omega}\frac{\partial \mathbf{P}^\omega}{\partial \mathbf{P}^s} = \frac{\partial G}{\partial \mathbf{P}^\omega}\mathbf{L}^{-1}, \tag{5.45}$$

after which the nullspace of this scaled constraint matrix needs to be computed.

Based on these discussions, one can see that the key point of implementing this scaling is to compute the inverse of scaling matrix, $\mathbf{L}^{-1}$. Since $\mathbf{L}$ is a diagonal matrix, the inverse is then easy to obtain.

5.7 Fixing some directions of homogeneous control points

Quite frequently, in CAD-based shape optimisation, control points of the shape need to be constrained in a part of the (x, y, z) directions. For instance, in the U-bend test case which will be presented in Chapter 8, the control points at the interfaces of fixed and free patches have to be fixed in certain directions to maintain G_1 continuity.

The constraints of control point position and weights are implemented in this research in a unified way using the homogeneous form of control points. During the optimisation, the following constraints can be applied now:

- Fixing control point in all directions
- Fixing control point in some directions of (x, y, z)
- Fixing control point weight

For a homogeneous form control point $\mathbf{P}^{\omega} = (\omega x, \omega y, \omega z, \omega)$, if the weights are fixed then it is simple to fix some directions of (x, y, z). We simply need to fix $(\omega x, \omega y, \omega z)$ correspondingly. If the weights and all directions are free, then nothing need to be done for the control points. However, if weights are free but some directions need to be fixed, then more attention should be paid to handle this. For example, if one wants to fix the x direction but make weights free, then one cannot do this by fixing ωx only. This is because that there is a projection process from 4-D space to 3-D space after geometry perturbation, as presented in Section 5.2. If ωx is fixed but ω is free, then after projection, the value of x will be different from the initial x. Actually, in this case, ωx should be divided using the initial weights ω^0 when performing the projection, such that the value of x doesn't change.

To sum up, for the homogeneous form control point $\mathbf{P}^{\omega} = (\omega x, \omega y, \omega z, \omega)$, one should follow the Algorithms 5.3 to fix a part of the (x, y, z) directions (taking the x direction

as example):

Algorithm 5.3: Fixing control point in x direction

```
if Weight of this control point is fixed then
    simply fix ωx;
    project from 4-D sapce to 3-D space using ω;
else
    fix ωx;
    project from 4-D sapce to 3-D space using the initial weight of this control
     point ω^0;
end
```

Other directions, such as y and z, can be fixed accordingly.

5.8 The application of STEP file in CAD-based shape optimisation

In this work, STEP file is used as both the input and output of the CAD-based shape optimisation framework. Here we introduce the application of STEP file.

5.8.1 Introduction to STEP file

STEP is a comprehensive ISO standard (ISO 10303) that describes how to represent and exchange digital product information among different systems [233–235]. Its scope is much boarder than the existing CAD data exchange formats, notably the IGES [104, 236]. It can be read by most, if not all, CAD systems. Therefore, STEP file has been widely used in academia and industry [237–245].

Normally, a STEP file is exported by users after modelling a geometry in CAD systems, such as CATIA[6], NX[7], or FreeCAD[8]. STEP file is based on NURBS curves and surfaces, such that we can represent NURBS surfaces according to equation (2.1) using information extracted from STEP file, and then use them in shape optimisation.

[6] https://www.3ds.com/products-services/catia
[7] https://www.plm.automation.siemens.com/en_gb/products/nx/
[8] https://www.freecadweb.org/

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(
/* description */ (''),
/* implementation_level */ '2;1');
FILE_NAME(
...
...
FILE_SCHEMA (('CONFIG_CONTROL_DESIGN'));
ENDSEC;

DATA;
...
...
#54=(
BOUNDED_SURFACE()
B_SPLINE_SURFACE(2,1,((#150,#151),(#152,#153),(#154,#155),(#156,#157),(#158,
#159)),.UNSPECIFIED.,.F.,.F.,.F.)
B_SPLINE_SURFACE_WITH_KNOTS((3,2,3),(2,2),(0.,15.707963267949,31.4159265358979),
(0.,20.),.UNSPECIFIED.)
GEOMETRIC_REPRESENTATION_ITEM()
RATIONAL_B_SPLINE_SURFACE(((1.,1.),(0.707106781186548,0.707106781186548),
(1.,1.),(0.707106781186548,0.707106781186548),(1.,1.)))
REPRESENTATION_ITEM('')
SURFACE()
);
...
...
#149=CARTESIAN_POINT('',(0.,0.,0.));
#150=CARTESIAN_POINT('',(10.,0.,0.));
...
...
ENDSEC;
END-ISO-10303-21;
```

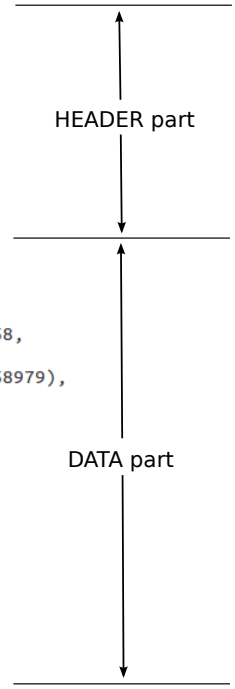


Figure 5.16: A part of STEP file corresponding to the half-cylinder.

A part of the STEP file of the half-cylinder in Fig. 5.7 is shown in Fig. 5.16. A STEP file usually consists of two main parts, namely HEADER and DATA part. Basic description information about the file, such as the file name and the name of software used to produce this file are described in the HERDER part, while the DATA part contains geometry and topology information. In this study, only information in the DATA part are utilised in shape optimisation.

Both geometry and topology information are read by the LOADSTEP module in NSPCC (see Section 5.1), although currently only geometry information, such as the position of control point, will be changed during optimisation. Topology information are kept unchanged in this study, but they provide essential information for successful shape optimisation.

5.8.2 The application of STEP file

Reading and writing the STEP file correctly are essential for this research. In-house STEP reader (LOADSTEP) and writer (WRITESTEP) supporting NUBRS have been developed in NSPCC, as shown in Fig. 5.1.

Reading geometry information from STEP file

In our work, the key geometry informations need to be read from STEP file are those used to build surfaces and curves, i.e. NURBS and B-splines if there are any present.

NURBS curve and surface

A NURBS curve entity in STEP file is shown in Fig. 5.17, indicated by the keywords 'BOUNDED_CURVE'. The following information required to represent the curve mathematically are provided:

- '#230': the entry ID of this NURBS curve.
- '2': the degree of B-spline curve.
- '(#424, ..., #428)': the list of control points of this curve.
- '(3, 2, 3)': the knot multiplicity.
- '(0., 15.70796, 31.41592)': the knot vector.
- '(1., 0.70710, 1., 0.70710, 1.)': weights of control points.

```
#230=(BOUNDED_CURVE()
B_SPLINE_CURVE(2,(#424,#425,#426,#427,#428),.UNSPECIFIED.,.F.,.F.)
B_SPLINE_CURVE_WITH_KNOTS((3,2,3),(0.,15.707963267949,31.4159265358979),
 .UNSPECIFIED.)
CURVE()
GEOMETRIC_REPRESENTATION_ITEM()
RATIONAL_B_SPLINE_CURVE((1.,0.707106781186548,1.,0.707106781186548,1.))
REPRESENTATION_ITEM('')
);
```

Figure 5.17: An example NURBS curve in STEP file.

A NURBS surface entity in STEP file is shown in Fig. 5.18, with parameters have similar meaning with NURBS curve. But now both information in u and v direction of the surface (see equation (2.1)) are listed.

```
#261=(
BOUNDED_SURFACE()
B_SPLINE_SURFACE(2,1,((#387,#388),(#389,#390),(#391,#392),(#393,#394),(#395,
#396)),.UNSPECIFIED.,.F.,.F.,.F.)
B_SPLINE_SURFACE_WITH_KNOTS((3,2,3),(2,2),(31.4159265358979,47.1238898038469,
62.8318530717959),(0.,100.),.UNSPECIFIED.)
GEOMETRIC_REPRESENTATION_ITEM()
RATIONAL_B_SPLINE_SURFACE(((1.,1.),(0.707106781186548,0.707106781186548),
(1.,1.),(0.707106781186548,0.707106781186548),(1.,1.)))
REPRESENTATION_ITEM('')
SURFACE()
);
```

Figure 5.18: An example NURBS surface in STEP file.

B-spline curve and surface

Compared to NURBS curves and surfaces, B-splines are simpler because they do not have rational weights. In a STEP file, they are presented in different form from those of NURBS, as can be seen from Fig. 5.19 and Fig. 5.20.

```
#166=B_SPLINE_CURVE_WITH_KNOTS('',1,(#497,#498),.UNSPECIFIED.,.F.,.F.,(2,
2),(28.,42.),.UNSPECIFIED.);
#167=B_SPLINE_CURVE_WITH_KNOTS('',1,(#499,#500),.UNSPECIFIED.,.F.,.F.,(2,
2),(42.,56.),.UNSPECIFIED.);
#168=B_SPLINE_CURVE_WITH_KNOTS('',1,(#501,#502),.UNSPECIFIED.,.F.,.F.,(2,
2),(42.,56.),.UNSPECIFIED.);
```

Figure 5.19: B-spline curve entities in STEP file.

```
#141=B_SPLINE_SURFACE_WITH_KNOTS('',5,5,((#142,#143,#144,#145,#146,#147),(#1
48,#149,#150,#151,#152,#153),(#154,#155,#156,#157,#158,#159),(#160,#161,#162
,#163,#164,#165),(#166,#167,#168,#169,#170,#171),(#172,#173,#174,#175,#176,#
177)),.UNSPECIFIED.,.F.,.F.,.U.,(6,6),(6,6),(0.,140.000009866),(0.,12.079215
0248),.UNSPECIFIED.) ;
```

Figure 5.20: A B-spline surface entity in STEP file.

Reading topology information from STEP file

Topology information is also very important in describing a geometry. Although we do not change the topology during the shape optimisation, we still need the topology information to give more insight on the geometry.

The topology information of one surface in the half-cylinder shown in Fig. 5.7 is listed in Fig. 1.4. As it can be observed, the topology information clearly tells how the

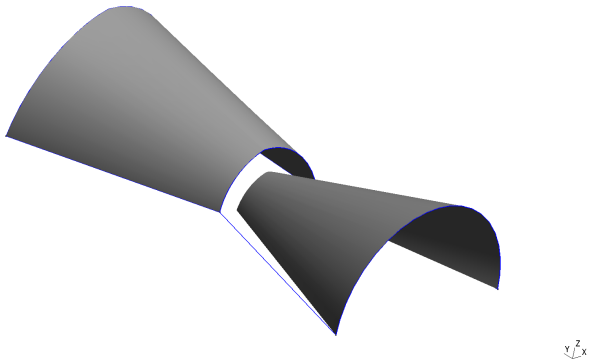


Figure 5.21: The gap between a surface and its boundary curves.

geometry is constructed. For example, we can find the curves forming the boundary of this surface. This is essential, because when a surface is updated, the boundary curves should also be updated accordingly. Otherwise, there will be a gap in the geometry, as illustrated in Fig. 5.21.

The LOADSTEP module in the NSPCC can parse the STEP file to read both geometry and topology information. In addition, it supports both the NURBS form (Fig. 5.17 and Fig. 5.18) and B-splines form (Fig. 5.19 and Fig. 5.20). Therefore, this reader can support a wide range of geometries.

Writing updated information into STEP file

To use STEP file successfully in the shape optimisation, it is essential to update the STEP file correctly. For example, write the updated position and weight of control points back. The WRITESTEP module in NSPCC (see Fig. 5.1) is devoted to handle everything about updating the STEP file.

Since NURBS is the generalisation of B-splines, in NSPCC we use NURBS to describe both NURBS and B-splines entries, i.e. describe them in a unified form as shown in Fig. 5.18 and Fig. 5.17. Therefore, there are two situations need to dealt with. The first one is from NURBS entry to NURBS entry, while the other one is from B-splines entry to NURBS entry.

For the former case, it is easy to write the updated information into STEP file. One only needs to locate the informations that need to be updated, such as the coordinates or weights control points, and then update.

For the second case, it is more complicated, because one firstly needs to convert the B-splines entry to the NURBS form and then update. To this end, all the weights of B-splines control points are set as 1.0. For example, a B-spline curve in original STEP file shown in Fig. 5.22 is converted to the NURBS form in the updated STEP file, as illustrated in Fig. 5.23.

```
#131=B_SPLINE_CURVE_WITH_KNOTS('',5,(#143,#144,#145,#146,#147,#148,#149,
#150,#151,#152,#153,#154,#155),.UNSPECIFIED.,.F.,.F.,(6,1,1,1,1,1,1,6),
(0.,0.106770833333333,0.234375,0.361979166666666,0.489583333333334,0.6171875,
0.744791666666668,0.872395833333334,1.),.UNSPECIFIED.);
```

Figure 5.22: A B-spline curve in original STEP file of M6 wing.

```
#131=(BOUNDED_CURVE()B_SPLINE_CURVE(5,(#143,#144,#145,#146,#147,#148,#149,#150,#151,#152,#153,#154,#155),.UNSPECIFIED.,.F
.,.F.)B_SPLINE_CURVE_WITH_KNOTS((6,1,1,1,1,1,1,6),(0.,0.106770833333333,0.234375,0.361979166666666,0.489583333333334,0.
6171875,0.744791666666668,0.872395833333334,1.),.UNSPECIFIED.)CURVE()GEOMETRIC_REPRESENTATION_ITEM()RATIONAL_B_SPLINE_CUR
VE((1.000000000000000,1.000000000000000,1.000000000000000,1.000000000000000,1.000000000000000,1.000000000000000,1.0000000
00000000,1.000000000000000,1.000000000000000,1.000000000000000,1.000000000000000,1.000000000000000,1.000000000000000))REP
RESENTATION_ITEM(''));
```

Figure 5.23: The converted NURBS curve in updated STEP file of M6 wing.

5.8.3 The connection between patches

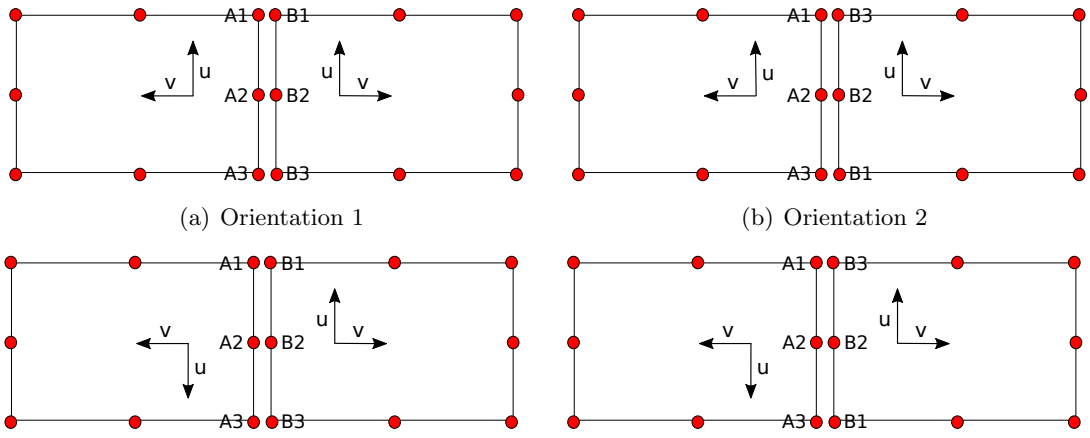
To impose geometric continuity constraint as presented in Section 5.4, the connection between patches are needed, such as which edge are the common edge of two adjacent patches, in which orientation. There are 16 different orientations between two adjacent patches, Figure 5.24 illustrates 4 of them. These connection informations are worked out based on the information extracted from the STEP file, by computing the distance different edges. For example, the distance between 4 edges of patch 1 and 4 edges of patch 2. If the distance between two edges are smaller than the chosen tolerance, these two edges are considered to be connected together.

The distance between edges is computed in following way:

1. Compute the distance between 4 corner points, i.e. A1-B1, A1-B3, A3-B1, A3-B3 in Fig. 5.24.

2. Compute the distance between 2 middle points, i.e. A2-B2 in Fig. 5.24.

3. The summation of these five values is used as the distance of two edges.



(a) Orientation 1 (b) Orientation 2

(c) Orientation 3 (d) Orientation 4

Figure 5.24: Different orientations of two patches sharing one common edge.

5.9 Shape optimisation framework based on NSPCC

During this research, a CAD-based shape optimisation framework (see Fig. 1.3) is developed, coupling the NSPCC CAD kernel, a flow solver, an adjoint solver, and a gradient-based optimiser. All the optimisation test cases presented in this thesis (see Chapter 7, Chapter 8 and Chapter 9) are performed using this framework.

The sensitivity computation mode of the optimisation framework is illustrated in Fig. 5.25. As introduced previously, the flow sensitivity $\frac{dJ}{d\mathbf{X}_s}$ is computed using the adjoint method, which is implemented using AD in reverse mode. Regarding the CAD sensitivity, currently this is computed by using forward differentiation inside NSPCC.

This optimisation loop holds several advantages. Firstly, due to the geometric property of NURBS, smooth geometry can be obtained without smoothing procedure which is required in the node-based method. Secondly, both the initial and optimised geometries exist in STEP format, which can be handled by almost every CAD software. This facilitates the linkage and information exchange between this optimisation chain and other software for post-processing or manufacturing, which is especially important in multidisciplinary design optimisation (MDO). Thirdly, this optimisation loop is fully automatic, namely no user input is required during the optimisation process. Once the user finishes the set up and starts the optimisation, everything will be automatically performed until the optimised geometry is returned to the user. Actually, the estimate of the number of test points beforehand and the application of effective rank enhance this automatic property of the framework. In addition, this optimisation loop is modular, which means

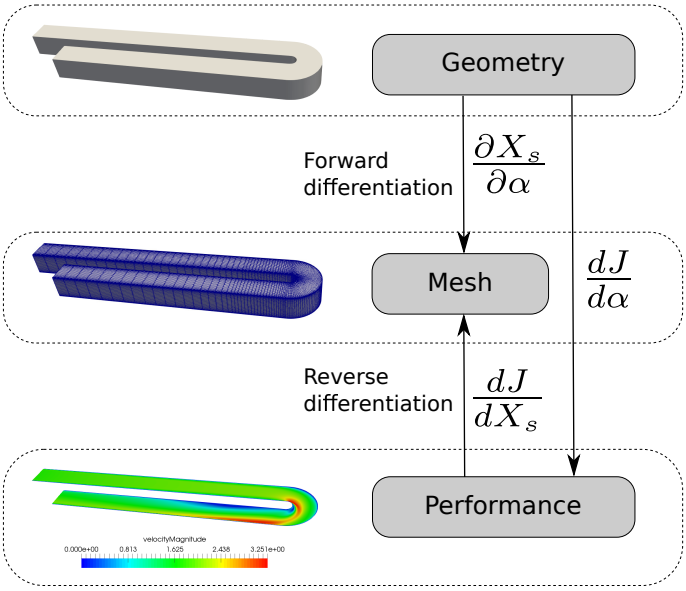


Figure 5.25: Sensitivity computation in CAD-based shape optimisation framework.

the user can replace some ingredients to achieve different functionalities. For example, by replacing the flow/adjoint solver with structural solver, the optimisation framework can also perform structural optimisation. Another example is that if different optimiser is used, the framework can achieve different optimisation performances (see Chapter 8). Finally, because the STEP file is available in each optimisation iteration, it is convenient for the user to check intermediate results during the optimisation process and regenerate the mesh, if necessary.

NSPCC plays a key role in the CAD-based shape optimisation chain. The findings of this chapter, and the development performed during this research actually make both NSPCC and the optimisation framework more powerful and automatic.

5.10 Summary

This chapter presents a detailed discussion on the NURBS-based parametrisation with continuity constraints, or the NSPCC approach, developed at QMUL.

Firstly, a general introduction to the overall workflow of NSPCC kernel and its five modules are given. Secondly, the extension of the in-house CAD kernel from B-splines to NURBS is presented. Thirdly, the point inversion algorithm or mesh projection process is discussed in detail. This process is the link between geometry and mesh, thus it is crucial

for the CAD-based shape optimisation. Then, the method to deal with continuity is introduced, along with the key points to have a good performance, such as the nullspace computation, the effective rank, the required number of test points, and the continuity recovery steps. Finally, some implementation details, such as the application of STEP file, and how to fix a part of directions, are presented.

The NSPCC CAD kernel is lightweight while powerful, and can be effectively coupled with different solvers to perform gradient-based shape optimisation, as will be demonstrated in following chapters.

Chapter 6

Validation

In gradient-based optimisation, one of the most important aspects is to compute the gradient accurately, because the gradient gives the direction along which the design can be improved. In the NSPCC approach, it is crucial to construct the constraint matrix correctly to impose the geometric continuity.

This chapter firstly introduces the computation of NURBS derivatives based on AD and verifies them in Section 6.1. Then Section 6.2 describes the verification of geometric gradients obtained through AD in NSPCC. The validation of underlying principle of NSPCC related to continuity are then presented, including the validation of constraint matrix reported in Section 6.3, and G_1 continuity recovery shown in Section 6.4.

The verification and validation in this chapter are devoted to demonstrate the effectiveness of NSPCC, therefore all tests only use geometric information. Flow and adjoint solvers are not involved.

6.1 Calculation and verification of NURBS derivatives

The NURBS derivatives consist of two parts, namely the derivatives of NURBS surface point coordinates w.r.t. parametric coordinates (u, v) and control points. Both parts are needed in the NSPCC approach. To be precisely, the derivatives of NURBS surface w.r.t. parametric coordinates (u, v) are used in two ways. First, $\mathbf{S}_u$, $\mathbf{S}_v$, $\mathbf{S}_{uu}$, $\mathbf{S}_{vv}$, $\mathbf{S}_{uv}$ are required in the mesh projection presented in Section 5.3. Second, $\mathbf{S}_u$, $\mathbf{S}_v$ are needed in G_1 continuity computations introduced in Section 5.4.1. The derivatives of surface point w.r.t. control points, $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}}$, are needed in equation (5.27) to compute the sensitivity of object function w.r.t. design variables.

The NURBS derivatives have been analytically derived and used in shape optimisation previously [111, 112, 246–249]. However, it is error-prone to derive the expression of the derivative and needs a lot of human effort, due to the complex rational representation of NURBS. The NURBS derivatives can also be computed using FD, but FD is sensitive to the step size as discussed in Section 3.4.1. Therefore, FD is only used to compare results, but not actually utilised in NSPCC. In addition, Xu et al. [48] computed $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}}$ in equation (5.27) according to the definition of NURBS surface. However, if weights are included into the design space, this term will become $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^w}$, which cannot be computed in the same way. Therefore, a proper method to compute this term is required.

6.1.1 Computing NURBS derivatives using AD

As introduced in Section 3.4.4, AD is easy to use, and the results of AD are accurate to the machine precision level. Therefore, in this work AD is employed to calculate both kinds of NURBS derivatives.

The forward mode AD is suitable for cases which have more dependent variables than independent variables, while the reverse mode could be used when dependent variables is less than independent variables. In NSPCC, when computing the derivatives of surface point w.r.t. parametric coordinates, because the number of surface points is the same as parametric coordinates, both modes are the same in terms of computational costs. The forward mode AD is implemented due to its simplicity.

In the NSPCC approach, the information about the geometry are read from the STEP file as introduced in Section 5.8. These informations are then used to rebuild the NURBS curves and surfaces. Algorithms used for this are taken from [51], and the subroutines are developed in house. The AD tool Tapenade is applied in forward mode to these subroutines to obtain the differentiated subroutines, which can be called to compute NURBS derivatives.

6.1.1.1 Computing the first order NURBS derivatives using AD

There is a subroutine computing the NURBS surface points in NSPCC:

```
1   S(u, v, cp_w, s)
```

where u and v are parametric coordinates, cp_w is the homogeneous control point, s is surface point. To obtain the first order NURBS derivatives, one simply needs to run the AD tool Tapenade in forward mode to this subroutine, specifying u, v, cp_w as inputs

and s as output. The differentiated subroutine is

```
S_d(u, ud, v, vd, cp_w, cp_w_d, s, sd)
```

In this subroutine, ud, vd and cp_w_d are seeds, their value should be given by the user to specify which derivative to compute. sd stores the derivatives. The derivatives $\mathbf{S}_u$, $\mathbf{S}_v$ and $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^\omega}$ can be computed using this subroutine. For example, $\mathbf{S}_u$ can be obtained with following codes:

```
! dS/du in forward mode
ud=1.0; vd=0.0; cp_w_d = 0
call S_d(u, ud, v, vd, cp_w, cp_w_d, s, sd)
dSdu=sd
```

$\mathbf{S}_v$ and $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^\omega}$ can be computed accordingly. For simplicity, the source codes are given in Appendix D.

6.1.1.2 Computing the second order NURBS derivatives using AD

For second order NURBS derivatives, one firstly obtains the following subroutine by running AD in forward mode, with u and v as inputs, s as output:

```
S_d(u, ud, v, vd, cp_w, s, sd)
```

Then apply AD in forward mode to this subroutine again, specifying u and v as inputs, sd as output. This results in the differentiated subroutine:

```
S_d_d(u, ud, udd, v, vd, vdd, cp_w, s, sd, sdd)
```

In this subroutine, udd and vdd are seeds for the second derivative, sdd stores the second derivative. To compute $\mathbf{S}_{uu}$, the differentiated subroutine should be called in the following way:

```
! dsds/dudu
ud=1.0; udd=1.0;
vd=0.0; vdd=0.0;
call S_d_d(u, ud, udd, v, vd, vdd, cp_w, s, sd, sdd)
dsds/dudu = sdd
```

The source codes to compute $\mathbf{S}_{vv}$, $\mathbf{S}_{uv}$ can be found in Appendix D.

Based on these discussions, one can see that there is no need to derive the analytic expression of the NURBS derivatives if using AD. One simply needs to apply AD to the very basic subroutine S(u, v, cp_w, s).

6.1.2 Verification of NURBS derivatives

Two geometries are utilised to verify the NURBS derivatives. The first one is the half-cylinder shown in Fig. 5.7, the other one is the ONERA M6 wing [179]. For both cases, a simple 6×6 grid for each surface is used, as shown in Fig. 6.1. Therefore, there are in total 72 surface nodes. The parametric directions are also given in Fig. 6.1.

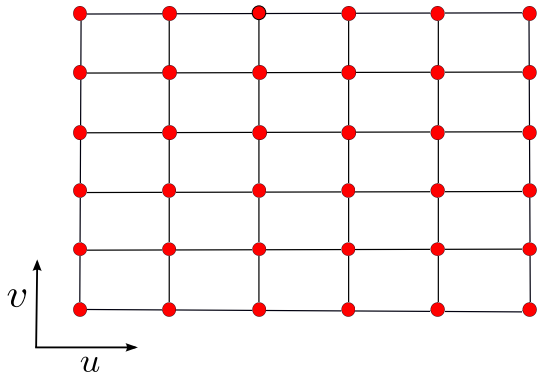
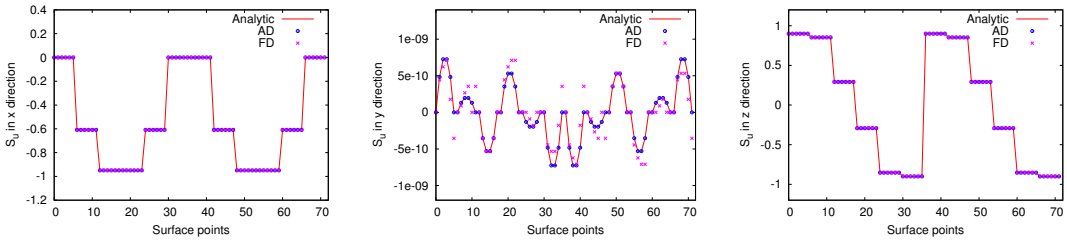


Figure 6.1: The 6×6 grid used for verification of NURBS derivatives.

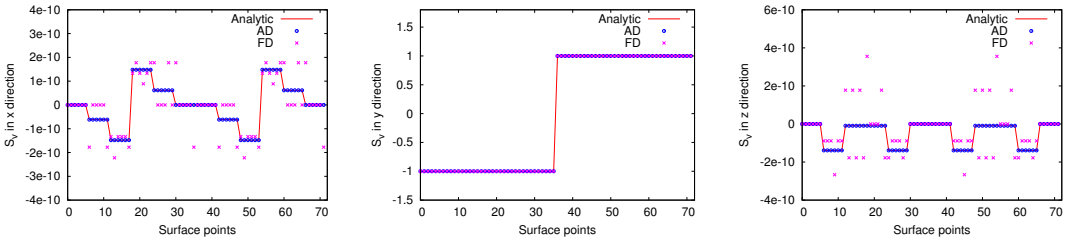
6.1.2.1 Verification using a half-cylinder geometry

For the half-cylinder geometry, the derivatives w.r.t. parametric coordinates are computed using three methods, namely analytic method, AD and FD. The step size of FD is 10^{-6}. The comparison of results are illustrated from Fig. 6.2 to Fig. 6.6:



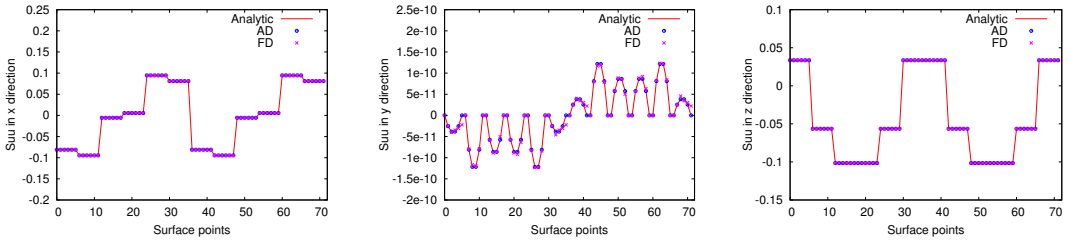
(a) x direction (b) y direction (c) z direction

Figure 6.2: $\mathbf{S}_u$ computed using analytic method, AD and FD (half-cylinder).



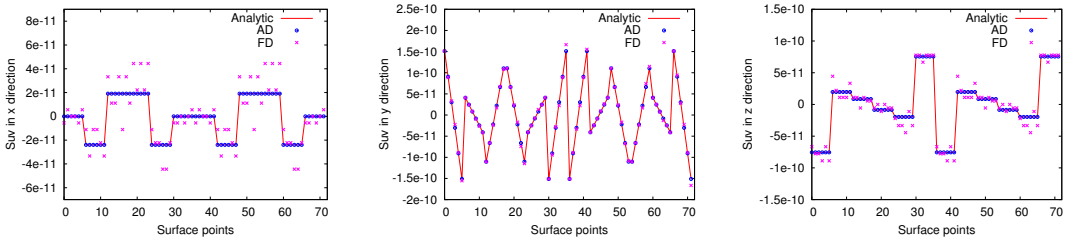
(a) x direction (b) y direction (c) z direction

Figure 6.3: $\mathbf{S}_v$ computed using analytic method, AD and FD (half-cylinder).



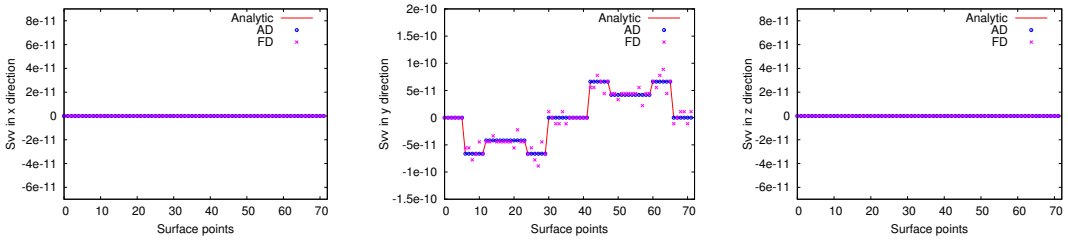
(a) x direction (b) y direction (c) z direction

Figure 6.4: $\mathbf{S}_{uu}$ computed using analytic method, AD and FD (half-cylinder).



(a) x direction (b) y direction (c) z direction

Figure 6.5: $\mathbf{S}_{uv}$ computed using analytic method, AD and FD (half-cylinder).



(a) x direction (b) y direction (c) z direction

Figure 6.6: $\mathbf{S}_{vv}$ computed using analytic method, AD and FD (half-cylinder).

These figures indicate that for each surface grid point, the derivatives w.r.t. parametric coordinates computed using analytic method and AD are exactly the same. This demonstrates that AD can provide very accurate derivatives. These figures also show that the results from FD match well with those of analytic method and AD.

For the derivatives w.r.t. homogeneous control points, for convenience the 9th mesh point i.e. point (2,3), and control point (3,1) on the first patch is picked to compare. The comparison of derivatives from AD and FD are given in Table 6.1. The second column lists the derivative of x coordinate of surface point w.r.t. control points coordinates. The third and fourth columns present the derivatives of y and z coordinates, respectively. A good agreement between the derivatives from AD and FD can be observed, thus the derivatives from AD are reliable.

Table 6.1: $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^\omega}$ computed using AD and FD (half-cylinder).

	x component	y component	z component
AD (ωx)	**0.076120462**990869	**0.0000000000000000**	**0.0000000000000000**
FD (ωx)	**0.076120463**014090	**0.0000000000000000**	**0.0000000000000000**
AD (ωy)	**0.000000000000000**	**0.076120462990869**	**0.0000000000000000**
FD (ωy)	**0.000000000000000**	**0.076120462990869**	**0.0000000000000000**
AD (ωz)	**0.000000000000000**	**0.0000000000000000**	**0.076120462**990869
FD (ωz)	**0.000000000000000**	**0.0000000000000000**	**0.076120462**570001
AD (ω)	**-0.703261**381049245	**0.0000000000000000**	**-0.291300**392239918
FD (ω)	**-0.703261**328282341	**0.0000000000000000**	**-0.291300**370339138

6.1.2.2 Verification using the ONERA M6 wing

Since the half-cylinder geometry contains only NURBS curves and surfaces, it would be better if a geometry consists of B-spline curves and surfaces can be used for further verification. To this end, the ONERA M6 wing approximated with B-splines [250] is used. The geometry and parametric directions are shown in Fig. 6.7.

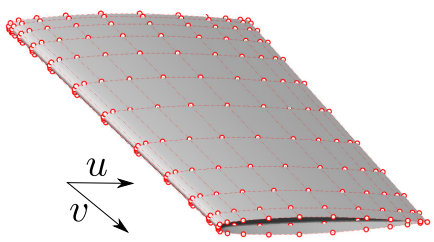
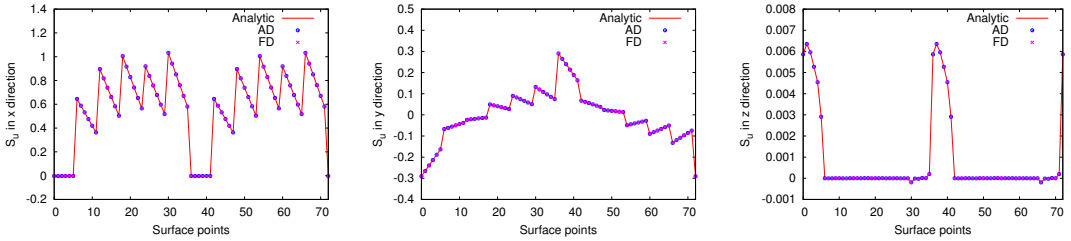


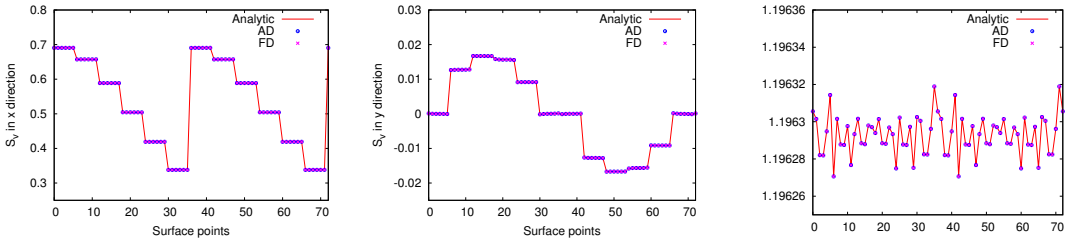
Figure 6.7: ONERA M6 wing and the (u, v) direction.

Similar to the half-cylinder case, for the derivatives w.r.t. parametric coordinates, three methods are applied to this geometry, namely analytic method, AD and FD. The results of $\mathbf{S}_u, \mathbf{S}_v, \mathbf{S}_{uu}, \mathbf{S}_{uv}, \mathbf{S}_{vv}$ are illustrated from Fig. 6.8 to Fig. 6.12. As can be seen, the derivatives obtained from three methods match well with each other.



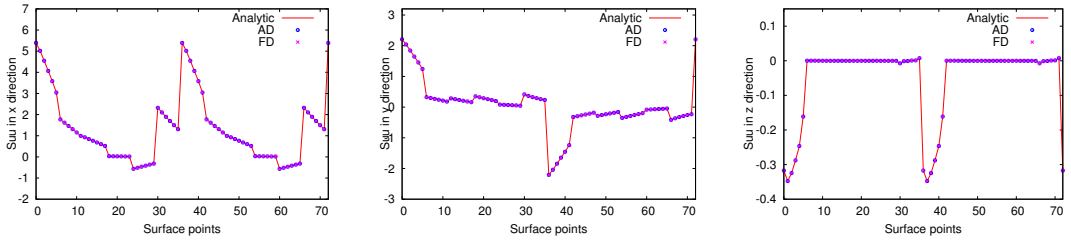
(a) x direction (b) y direction (c) z direction

Figure 6.8: $\mathbf{S}_u$ computed using analytic method, AD and FD (M6 wing).



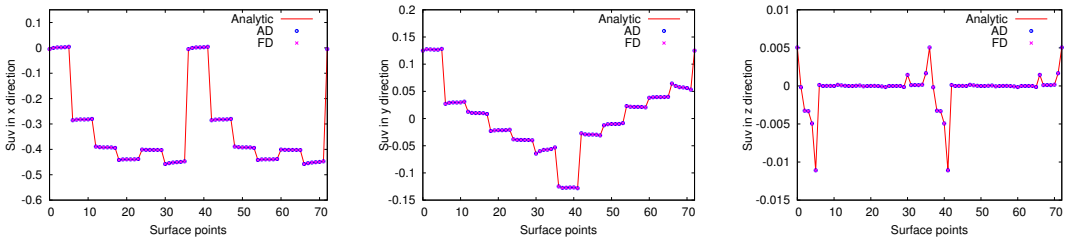
(a) x direction (b) y direction (c) z direction

Figure 6.9: $\mathbf{S}_v$ computed using analytic method, AD and FD (M6 wing).



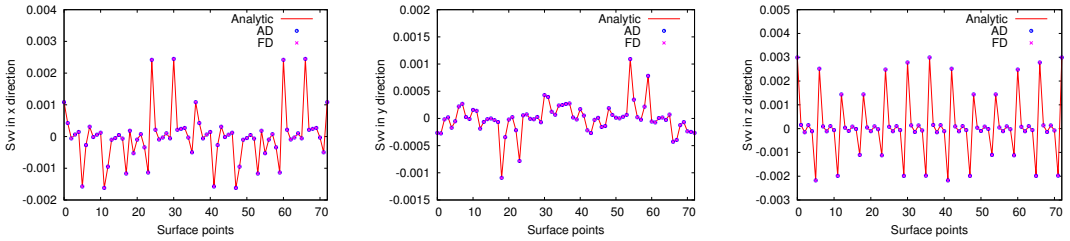
(a) x direction (b) y direction (c) z direction

Figure 6.10: $\mathbf{S}_{uu}$ computed using analytic method, AD and FD (M6 wing).



(a) x direction (b) y direction (c) z direction

Figure 6.11: $\mathbf{S}_{uv}$ computed using analytic method, AD and FD (M6 wing).



(a) x direction (b) y direction (c) z direction

Figure 6.12: $\mathbf{S}_{vv}$ computed using analytic method, AD and FD (M6 wing).

For the derivatives w.r.t. the homogeneous control point, the surface point $(3, 2)$ and control point $(7, 5)$ on the first patch is chosen to compare. The comparison of AD and FD derivatives are listed in Table 6.2, which illustrates a good agreement. More details of this case and the optimisation results will be discussed in Chapter 9.

Table 6.2: $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}^\omega}$ computed using AD and FD (ONERA M6 wing).

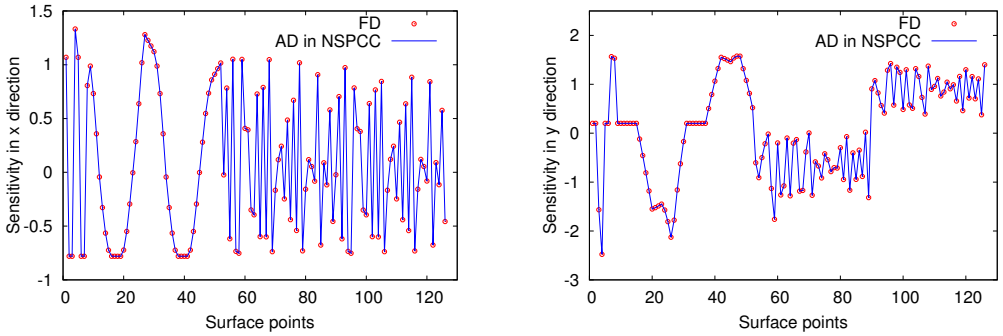
	x component	y component	z component
AD (ωx)	**0.113965435**854303	**0.000000000000000**	**0.000000000000000**
FD (ωx)	**0.113965435**820838	**0.000000000000000**	**0.000000000000000**
AD (ωy)	**0.000000000000000**	**0.113965435854**303	**0.000000000000000**
FD (ωy)	**0.000000000000000**	**0.113965435855**533	**0.000000000000000**
AD (ωz)	**0.000000000000000**	**0.000000000000000**	**0.113965435**854303
FD (ωz)	**0.000000000000000**	**0.000000000000000**	**0.113965435**876349
AD (ω)	**-0.039975**400313640	**0.003967185**875772	**-0.027267**21733905
FD (ω)	**-0.039975**395761291	**0.003967185**423528	**-0.027267**18620829

To sum up, we can conclude that the NURBS derivatives computed using AD in the NSPCC approach are reliable.

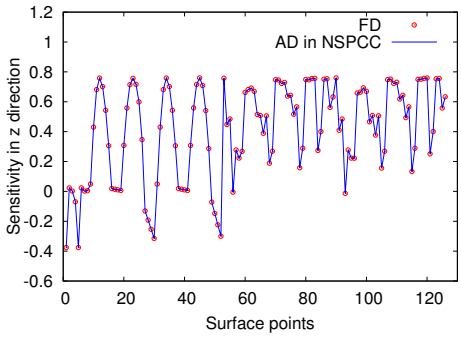
6.2 Verification of CAD sensitivity

In this section, the CAD sensitivities computed in NSPCC through AD, namely $\frac{\partial \mathbf{X}_s}{\partial \boldsymbol{\alpha}}$ in equation (5.26) are compared with the results from central finite differences method, for the purpose of verification.

6.2.1 Verification with half-cylinder



(a) x direction (b) y direction



(c) z direction

Figure 6.13: The comparison of $\frac{\partial \mathbf{X}_s}{\partial \alpha_1}$ computed by FD and AD in NSPCC for half-cylinder.

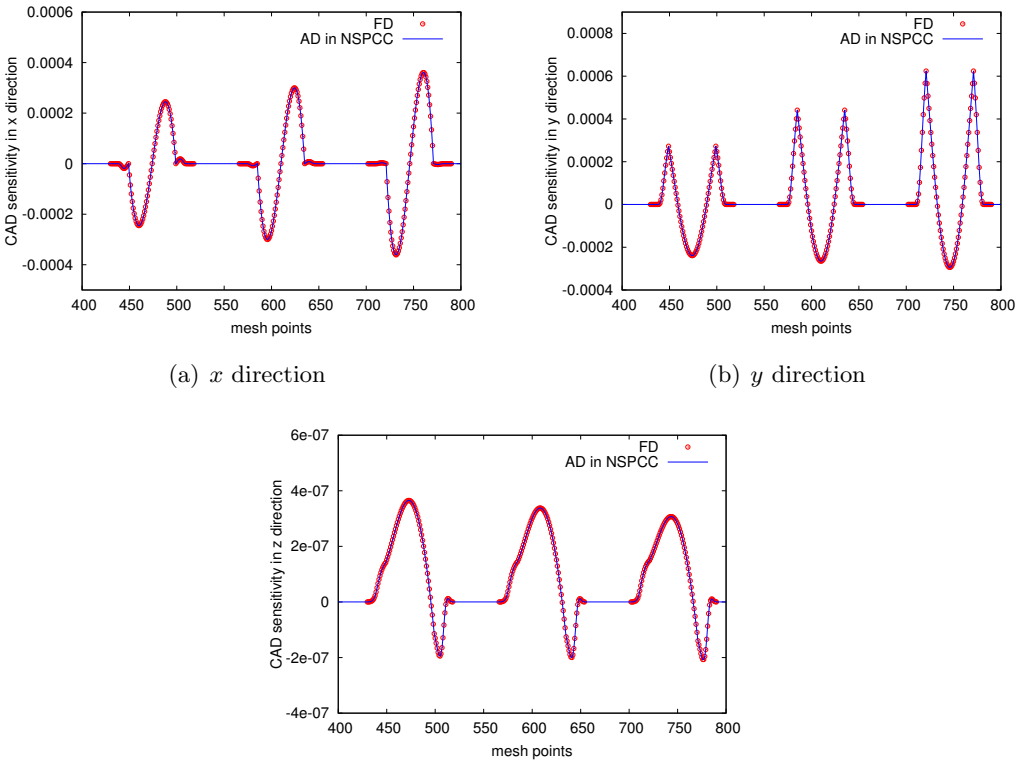
As a first case, the half-cylinder geometry illustrated in Fig. 5.7 is chosen. A surface mesh created in GMSH[1] with 126 mesh nodes (shown in Fig. 2.1) is utilised here. G_1 continuity is imposed at the common edge with 9 pairs of test points. All control points are free to move in the (x, y, z) directions and the weights are also free. In this case, there are 42 elements in the design vector $\boldsymbol{\alpha}$, we only show the comparison for α_1 for

[1] http://gmsh.info/

simplicity. The step size is 0.01. Figure 6.13 shows the comparison of the gradients in x, y and z directions. It can be seen that the sensitivity calculated by FD and AD have good agreement with each other. The gradient of other elements in $\boldsymbol{\alpha}$ can be verified accordingly.

6.2.2 Verification with U-bend

The CAD sensitivity verification is also performed for a U-bend cooling channel, which will be presented in Chapter 8 as a practical optimisation case. A hexahedral mesh with around 167K nodes is utilised for this case, and the step size is 0.001 for central finite differences. Again, the results of the first element in $\boldsymbol{\alpha}$ w.r.t. a part of surface mesh points are given in Fig. 6.14. It can be observed that the results from both methods exhibit high similarities, giving confidence in running optimisation for this cooling channel. More details of this case can be found in Chapter 8.



(a) x direction

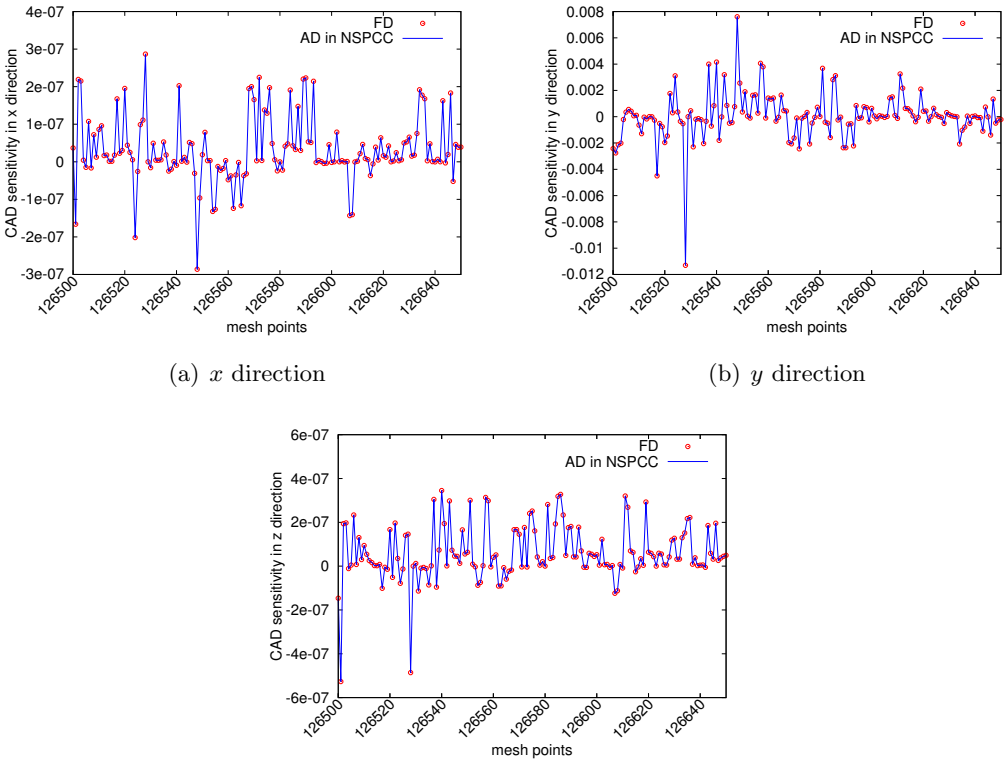
(b) y direction

(c) z direction

Figure 6.14: The comparison of $\frac{\partial \mathbf{X}_s}{\partial \alpha_1}$ computed by AD in NSPCC and FD for U-bend.

6.2.3 Verification with the ONERA M6 wing

As a third case, the ONERA M6 wing with shown in Fig. 6.7 is employed. All control points are free, including both position and weight. The step size is 0.001. The comparison of results for α_1 is presented in Fig. 6.15, which again demonstrates that the sensitivities obtained in NSPCC using AD are reliable.



(a) x direction

(b) y direction

(c) z direction

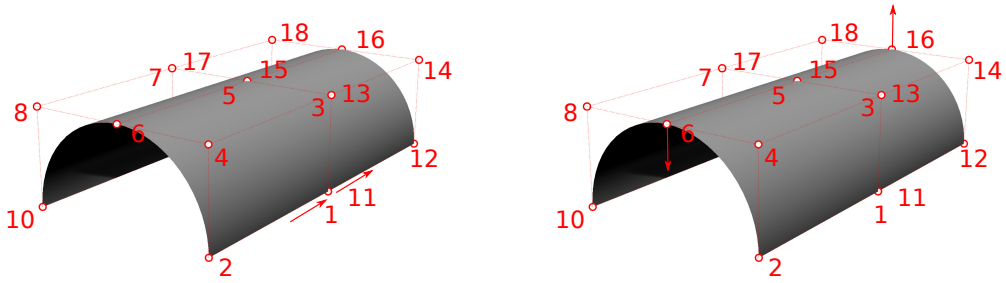
Figure 6.15: The comparison of $\frac{\partial \mathbf{X}_s}{\partial \alpha_1}$ computed by AD in NSPCC and FD for the ONEMA M6 wing.

6.3 Validation of constraint matrix

In the NSPCC approach, to ensure the continuity constraint are satisfied, the displacement of control points should lie in the nullspace of the constraint matrix. Therefore, it is essential to have correct constraint matrix. In this section, the validation of the constraint matrix will be presented.

According to Section 5.4.1, if $\delta\mathbf{P}$ lies in the nullspace of matrix $\mathbf{C}$, then $\mathbf{C}\delta\mathbf{P} = 0$. The

geometry used for validation is the half-cylinder shown in Fig. 5.7. For convenience, each control point is given a label as shown in Fig. B.1. In following tests, all the control points are free to move in the (x, y, z) directions.



(a) Perturbation of Test 1: in the nullspace of G_0 continuity

(b) Perturbation of Test 2: in the nullspace of G_1 continuity

Figure 6.16: The half-cylinder geometry and perturbations for Test 1 and Test 2.

Test 1: Imposing G_0 with $\delta\mathbf{P}$ in the null space

In this test, the G_0 continuity constraint is imposed with 7 test points along the common edge based on the results presented in Section 5.4.9, and all weights are free. Therefore, the dimension of constraint matrix $\mathbf{C}$ is 21×80, and the dimension of $\delta\mathbf{P}$ is 80×1. The perturbation used in this test is shown in Fig. 6.16(a), where the 1st and 11th control point are moved in $+y$ direction by the same amount. From Fig. 6.16(a) one can see that this perturbation will not break the G_0 continuity, thus it is in the nullspace.

To visualise the results, the transpose of $\mathbf{C}\delta\mathbf{P}$ is plotted as in Fig. 6.17. From this figure we can see that all the elements in $\mathbf{C}\delta\mathbf{P}$ are at the order of 10^{-15}. This matches with the expected result very well, therefore the matrix $\mathbf{C}$ is correct in this case.

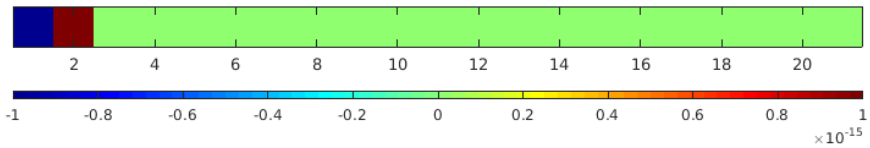


Figure 6.17: The transpose of $\mathbf{C}\delta\mathbf{P}$ in Test 1.

Test 2: Imposing G_1 with $\delta\mathbf{P}$ in the nullspace

In this test, the G_1 continuity constraint is imposed with 9 test points along the common edge, all weights are free. The dimension of matrix $\mathbf{C}$ is 54×80, and the

dimension of $\delta\mathbf{P}$ is 80×1. The displacements of control points are shown in Fig. 6.16(b), i.e. the 6th and 16th control points are moved toward -z direction and $+z$ direction for the same amount. After the perturbation, the 5th, 6th, 15th and 16th control point will still lie in one straight line, therefore this perturbation will not break the G_1 continuity [51]. So it is in the nullspace.

The transpose of $\mathbf{C}\delta\mathbf{P}$ is plotted in Fig. 6.18. As can be seen, all the elements in the result matrix are at the order of 10^{-15}.

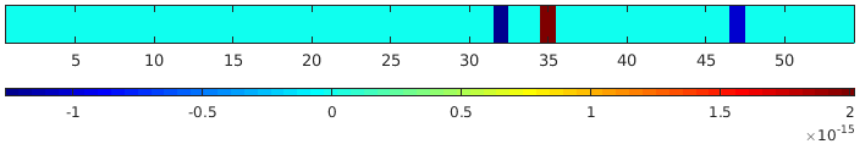


Figure 6.18: The transpose of $\mathbf{C}\delta\mathbf{P}$ in Test 2.

Based on these two tests, we can conclude that the computation of the constraint matrix in the NSPCC approach is reliable. Two more tests using perturbations which are not in the nullspace are given in Appendix B.

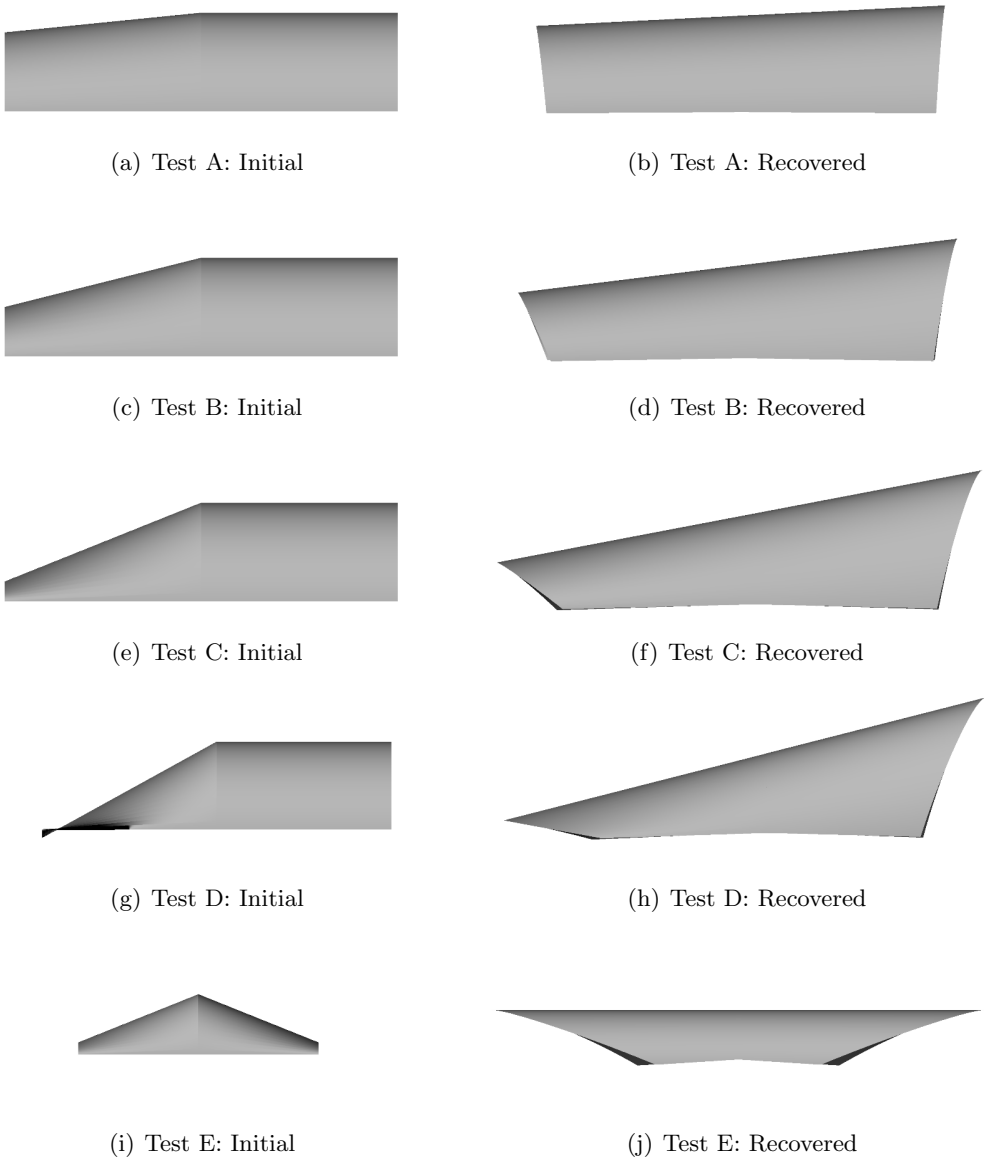
6.4 Validation of geometric continuity

In Section 6.3, the correctness of the constraint matrix constructed in NSPCC has been shown. However, $\mathbf{C}\delta\mathbf{P} = 0$ does not necessarily ensure the continuity is satisfied strictly if a finite-size step is taken, due to the non-linearity of constraints.

6.4.1 Validation of G_1 continuity recovery

The continuity recovery steps are necessary to recover the G_1 continuity during the optimisation. The idea is to take some steps normal to the nullspace to recover violated continuity. The details and basic process of the continuity recovery steps have been presented in Section 5.5. Here several small test cases are utilised to demonstrate the effectiveness of the G_1 recovery steps in recovering the G_1 continuity.

Based on the half-cylinder shown in Fig. 5.7, five deformed shapes are produced, as shown in Fig. 6.19. Initially, none of these shapes have G_1 continuity and the deviation from exact G_1 continuity increases from Test A to Test E. The initial G_1 deviation value of these five cases are shown in Table 6.3.



(a) Test A: Initial (b) Test A: Recovered

(c) Test B: Initial (d) Test B: Recovered

(e) Test C: Initial (f) Test C: Recovered

(g) Test D: Initial (h) Test D: Recovered

(i) Test E: Initial (j) Test E: Recovered

Figure 6.19: Initial (left) and recovered (right) geometries.

The G_1 deviation value of these five cases during the recovery process are shown in Fig. 6.20. It can be seen very clearly that the deviation value are reduced to smaller than 10^{-13} in just several steps for all five cases, indicating the G_1 continuity are obtained successfully. In general, the larger the initial deviation value is, the more recovery steps are needed. The only exception is Test D. The recovered shape are presented on the right of Fig. 6.19. These tests clearly demonstrate the effectiveness of the continuity recovery steps when applied to NURBS geometry.

Table 6.3: Initial G_1 deviation value of different half-cylinder case.

Test	Initial G_1 deviation value
Test A	2.407373060155627
Test B	14.490983918250024
Test C	34.746117585065129
Test D	60.215355030638726
Test E	124.643237299114830

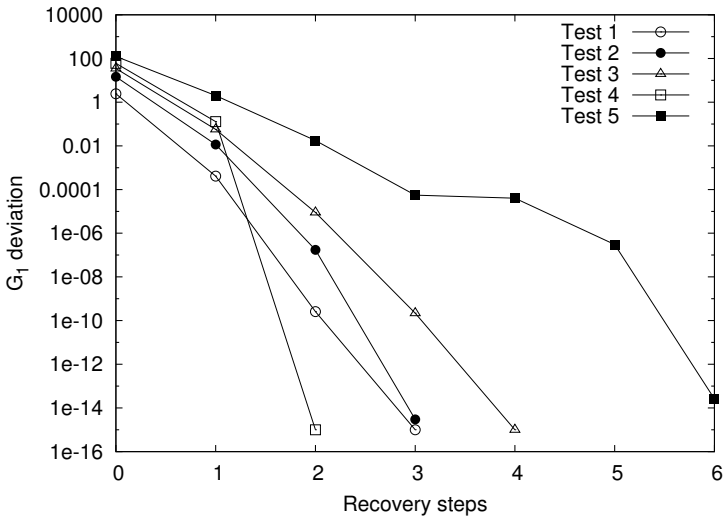


Figure 6.20: The G_1 deviation value during continuity recovery process.

6.4.2 Validation of imposing different constraints to different edges

As mentioned in Section 5.4.2, the NSPCC is developed to be able to impose different continuity constraints to different common edges in the geometry. In this section, a small test case is utilised to demonstrate this feature.

Figure 6.21 shows a long half cylinder geometry consists of four patches and their control points. There are three common edges in this geometry. Initially, all of these three common edges only have G_0 continuity. In this test, two common edges on the left and right side are imposed with the G_0 continuity, and the middle common edge is imposed with the G_1 continuity, as shown in Fig. 6.22(a). Again, the G_1 recovery steps are utilised to recovery the continuity. As can be seen from Fig. 6.23, after performing three G_1 continuity recovery steps, the G_1 deviation value is smaller than 10^{-12}. This means the middle edge now has G_1 continuity. The recovered geometry is illustrated in

Fig. 6.22(b), from which one can observe that the two common edges on the left and right side only hold G_0 continuity. This test shows that different continuity levels have been imposed to different edges successfully. The application of this functionality in practical CFD-based shape optimisation problem will be shown in Chapter 8.

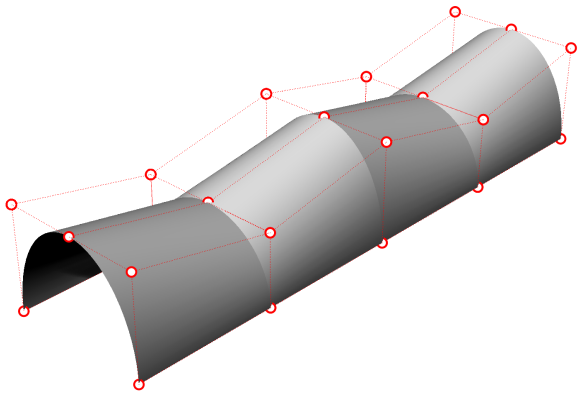
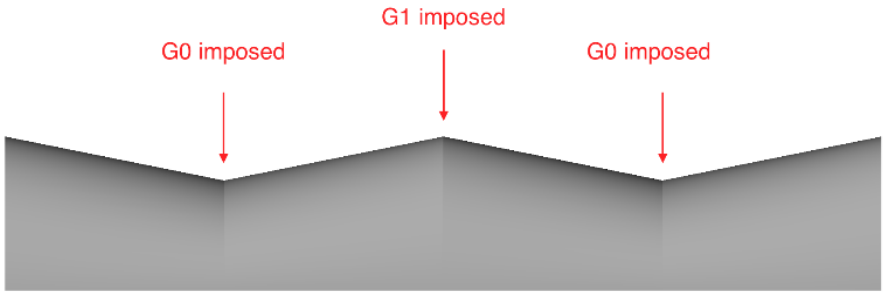


Figure 6.21: A long half-cylinder and its control points.



(a) The long half cylinder before recovery

(b) The long half cylinder after recovery

Figure 6.22: The continuity recovery results with different continuity levels imposed on different edges.

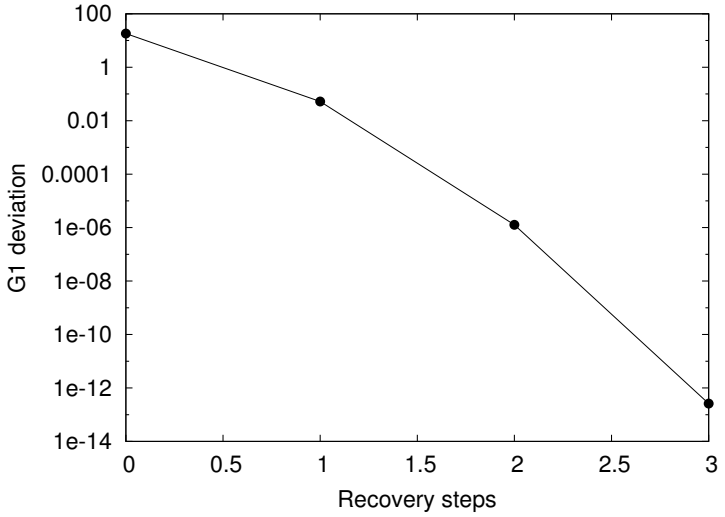


Figure 6.23: The G_1 deviation value during continuity recovery process for the long half-cylinder.

6.5 Summary

In this chapter, the NURBS derivatives which are necessary in NURBS-based shape optimisation are calculated using AD and verified. The results indicate that the calculations are reliable. Then the accuracy of geometric CAD sensitivity is also compared against FD and is found to be comparably accurate.

The underlying principle of NSPCC is then validated, including:

- The construct of constraint matrix.
- The continuity recovery step.
- Imposing different continuity constraint levels to different common edges in a geometry.

The tests performed in this chapter give confidence in the NSPCC approach, and lay a solid foundation for the application of NSPCC in practical gradient-based shape optimisations.

Chapter 7

Shape optimisation of an S-bend air duct

7.1 Introduction

The CFD-based shape optimisation has been applied to many areas, one of which is the automotive industry. For example, Taherkhani et al. [32, 33, 251] optimised the body of emergency response vehicles and police cars in order to reduce the drag, which helps to lower fuel consumptions.

In this chapter, the NSPCC approach is applied to an S-bend air duct segment from the automotive industry, provided by Volkswagen AG to the FlowHead research project[1]. The optimisation objective is to minimise the mass-averaged total pressure loss, which is defined as

$$J = \frac{\int_{inlet} p_{total}(\mathbf{u}_v \cdot \mathbf{n})dS - \int_{outlet} p_{total}(\mathbf{u}_v \cdot \mathbf{n})dS}{\int_{inlet}(\mathbf{u}_v \cdot \mathbf{n})dS}, \tag{7.1}$$

where p_{total} is the total pressure, $\mathbf{u}_v$ is the velocity vector, $\mathbf{n}$ is the normal direction and S is the cross section area.

The aim of this chapter is to show the feasibility of NSPCC in practical optimisation application, coupling with an incompressible solver. Besides, the constitution of design space and factors affecting the optimisation performance, such as cut-off value (see Section 5.4.7) and mesh density, are also explored and discussed.

[1] http://flowhead.sems.qmul.ac.uk/

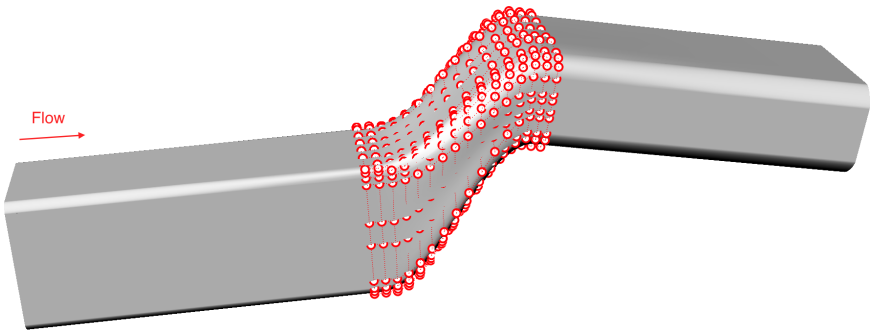


Figure 7.1: Control points of the middle section in the BRep.

7.2 Parametrisation

7.2.1 Geometry of the air duct

The initial shape of this S-bend duct is shown in Fig. 7.1. It consists of 30 surfaces, but only 8 of them in the middle part are allowed to move, while both inlet and outlet legs are fixed. The middle section consists of 4 larger and 4 smaller patches, each of which has 96 (16×6) and 64 (16×4) control points, respectively. Therefore, there are 640 control points on design surfaces in total, which are illustrated in Fig. 7.1 with hollow circles. One thing should be mentioned is that in this S-bend case, the design surfaces are not included in the definition of the objective function shown in equation (7.1), but only affect the flow. As a consequence, the term $\frac{\partial J}{\partial \boldsymbol{\alpha}}$ in equation (3.2) is 0.

Note that the duct bends upwards and sideways, thus it is asymmetric. Hence, the optimisation results should also be asymmetric.

7.2.2 Geometric continuity constraints

Geometric continuity constraints are required between adjacent deformable surfaces during the optimisation, so that continuous and smooth results can be obtained.

G_2 continuity is imposed between moveable surfaces and fixed patches (the inlet and outlet legs), as shown in Fig. 7.2. This is achieved by fixing the first three columns of control points on free patches near the interface [51]. As a result, 240 control points are fixed to ensure G_2 continuity, and 400 control points on these surfaces are free to move. Since each control point can move in the (x, y, z) directions, there are 1200 degrees of freedom (DoF) in total if weights of control points are frozen, and 1600 DoF if weights

are free. In this chapter, weights are not changed unless stated otherwise.

G_1 continuity is imposed at the interfaces of 8 deformable surfaces as illustrated in Fig. 7.2, by using the test point-based approach presented in Section 5.4.1. To achieve this, a number of pairs of test points need to be imposed at the interfaces, then a constraint matrix is built and its nullspace is computed using SVD. The movements of control points should lie in this nullspace to ensure the geometric continuity.

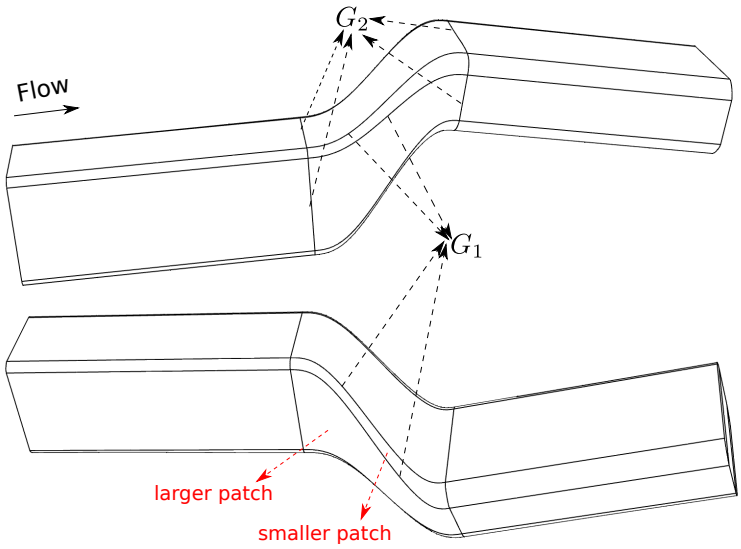


Figure 7.2: S-bend patches setup and geometric continuity.

As introduced in Section 5.4.9, the number of required test points can be estimated based on the knot vector, degree and number of control points along the common edge. In this S-bend case, the knot vector along the common edge is:

$$\{\underbrace{0,\ 0,\ \ldots,\ 0,\ 0}_{16},\ \underbrace{1,\ 1,\ \ldots,\ 1,1}_{16}\}$$

Therefore the non-zero knot intervals is 1. According to equation (5.34), the required number of test points should be

$$M_{testpoint} \geq (15+1) \cdot 1 f_T = 16 f_T.$$

The value of f_T could be between 1 and 1.5 according to the author's experience. In addition, since G_1 continuity is imposed, more test points are needed as shown in Section 5.4.9. Therefore, in this study, we impose 30 pairs of test points on each common edge to ensure the G_1 geometric continuity is satisfied among free patches. Figure 7.3 presents the relationship between the number of non-zero singular value and test points, which

indicates that the number of non-zero singular values will not increase any further if more than 30 test points are imposed. This has good agreement with the estimate given by equation (5.34).

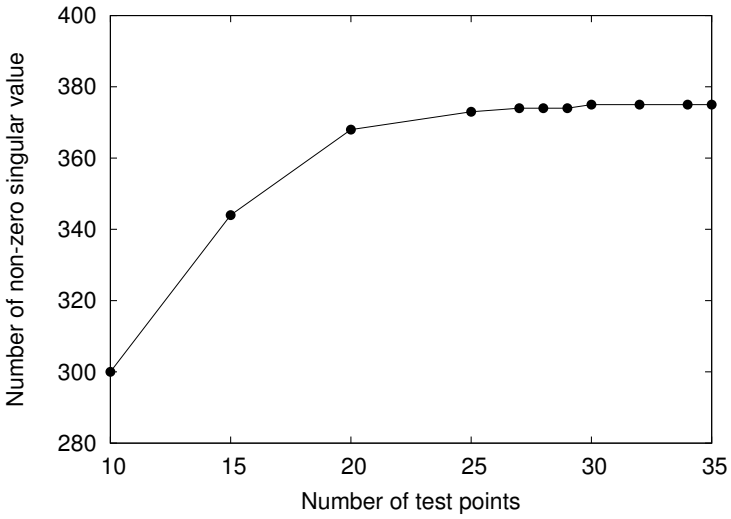


Figure 7.3: Relationship between the number of non-zero singular value and test point.

7.3 Solver and parameters

For most optimisations carried out in this chapter, a mesh with 41,044 hexahedron and 43,848 nodes is used, as shown in Fig. 7.4. An exception is in Section 7.6.4, where a finer mesh is utilised to investigate the effect of mesh density on optimisation results.

Main parameters for optimisation cases in this Chapter (except results in Section 7.6.4) are as following:

- The Reynolds number is 300, based on the height of the duct
- Mesh: 41,044 hexahedron with 43,848 nodes, as shown in Fig. 7.4
- Boundary conditions:
  - Inlet velocity is 0.1 m/s
  - Zero back pressure
  - Non-slip walls

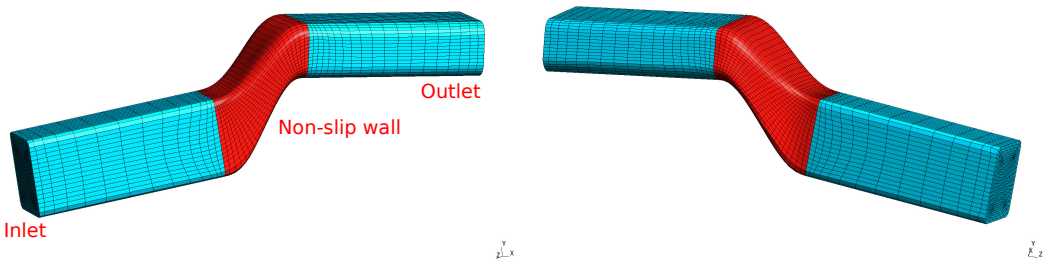


Figure 7.4: Hexahedral mesh of S-bend air duct.

Since this is an incompressible laminar case, the in-house flow and adjoint solver GPDE introduced in Sections 4.2.1 and 4.3.2 is employed. To be more specific, the flow solver inside GPDE is used to provide the flow solutions and the discrete adjoint solver is employed to calculate the sensitivity of the objective function w.r.t. surface node coordinates, $\frac{dJ}{d\mathbf{X}_s}$. In addition, the spring analogy technique provided in GPDE is used to deform the volume mesh, which is performed after the surface mesh deformation in the NSPCC.

The validation of GPDE solver for this S-bend air duct has been presented in Section 4.2.1, thus will not be repeated here. The interested reader could refer to those contents.

7.4 Gradient verification

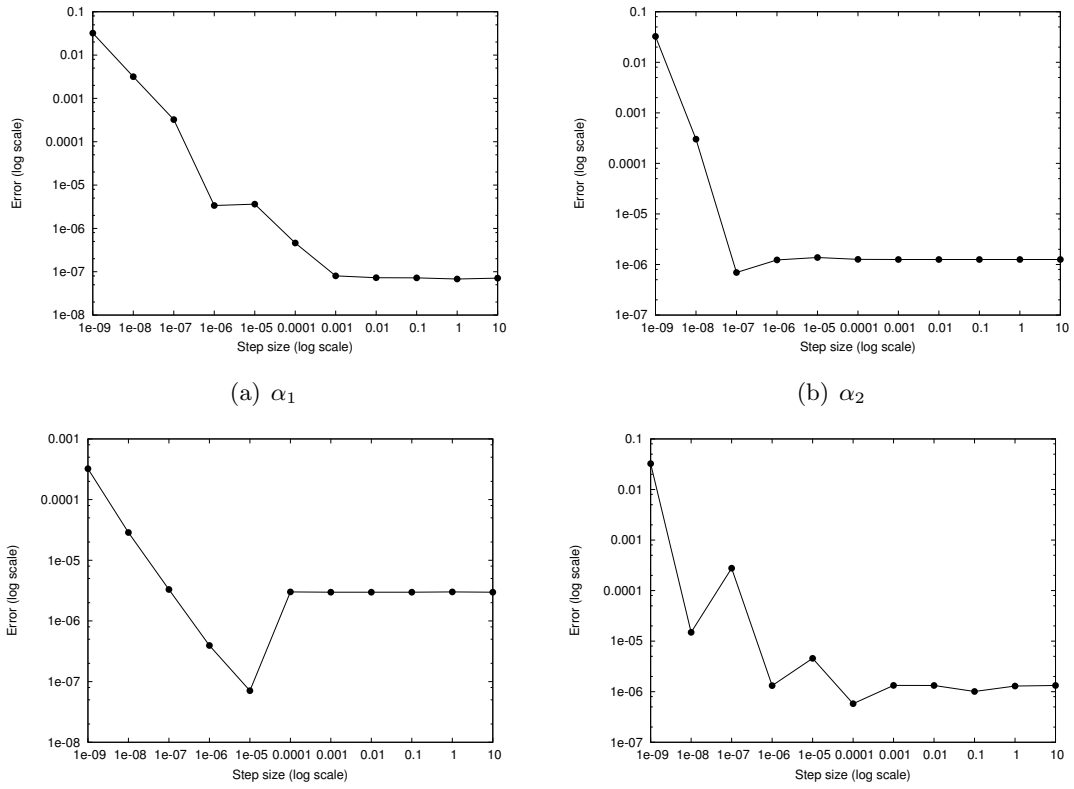
In this S-bend shape optimisation case, the derivatives of the objective function with respect to the design variables are:

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{dJ}{d\mathbf{X}_s}\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}}\frac{\partial \mathbf{P}}{\delta \boldsymbol{\alpha}} = \frac{dJ}{d\mathbf{X}_s}\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}}\mathrm{Ker}(\mathbf{C}), \tag{7.2}$$

where $\mathbf{P}$ is control points, $\boldsymbol{\alpha}$ is the vector of design variables, i.e. the linear combinational coefficients of deformation modes, $\frac{dJ}{d\mathbf{X}_s}$ is provided by the adjoint solver, $\frac{\partial \mathbf{X}_s}{\partial \mathbf{P}}$ is computed by using AD inside NSPCC, $\mathrm{Ker}(\mathbf{C})$ is the nullspace of the constraint matrix as introduced in Section 5.4.4. Here control points can move in the (x, y, z) directions.

The derivative $\frac{dJ}{d\boldsymbol{\alpha}}$ computed from equation (7.2) is then compared with the one given by central finite differences introduced in Section 3.4.1. For simplicity, the derivatives of first 4 design variables in the design vector $\boldsymbol{\alpha}$ are compared. The differences between results from two methods are illustrated in Fig. 7.5, which shows clearly that the derivatives from equation (7.2) have nice agreement with those from FD. More specifically, by

carefully choosing step size, the differences between these two derivatives are about 10^{-7} to 10^{-6}, which are the normal accuracy of FD. Figure 7.5 also indicates that for the first four design variables, the proper step sizes are different. This further demonstrates that FD suffers from choosing a proper step size, as mentioned in Chapter 3.



(a) α_1 (b) α_2

(c) α_3 (d) α_4

Figure 7.5: Difference between derivatives computed by using AD and FD.

7.5 Shape deformation modes in nullspace

As discussed in Section 5.4, because of the rounding errors in numerical computing, the nullspace used in practice is not the theoretically exact nullspace, but the numerical nullspace. The columns of the numerical nullspace will then constitute the design space through a linear combination. It would be helpful to plot the deformation modes in different areas of the numerical nullspace, as illustrated in Fig. 7.6.

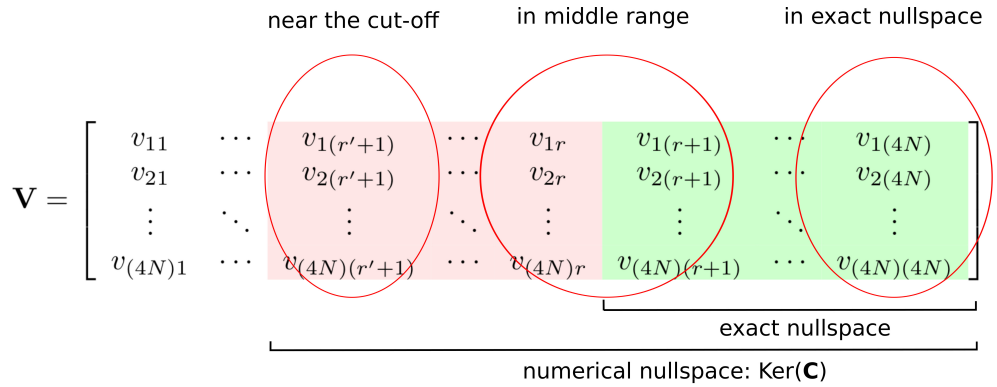


Figure 7.6: Different areas of the numerical nullspace.

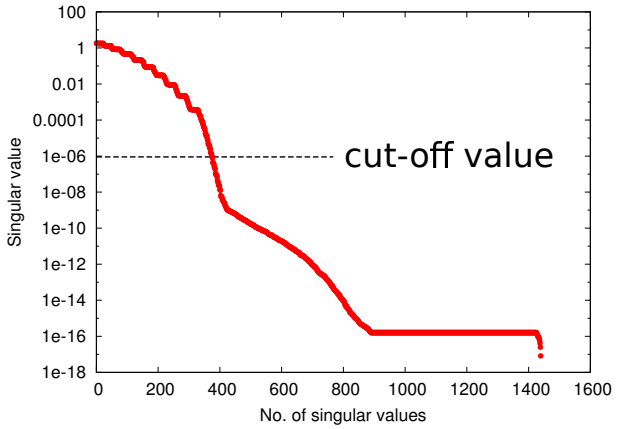


Figure 7.7: Singular value distribution of S-bend case.

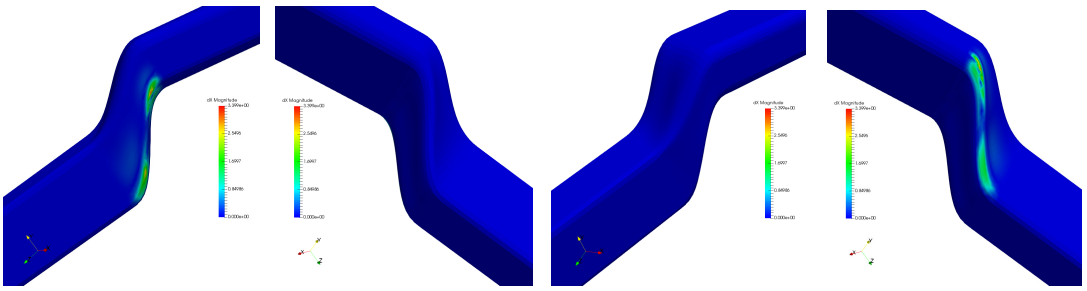
The numerical rank r' is 375 when the cut-off value σ_C is 10^{-6}, as illustrated in Fig. 7.7 where the singular value distribution is shown. Therefore, the dimension of Ker$\mathbf{C}$ is 1600×1225 and the size of $\boldsymbol{\alpha}$ is 1225, which means there are 1225 deformation modes in the design space. The deformation can be computed as:

$$\Delta \mathbf{X}_s = \frac{\partial \mathbf{X}_s}{\partial \mathbf{P}} \cdot \underbrace{\begin{bmatrix} v_{1,1} & \cdots & v_{1,600} & \cdots & v_{1,700} & \cdots & v_{1,1225} \\ v_{2,1} & \cdots & v_{2,600} & \cdots & v_{2,700} & \cdots & v_{2,1225} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v_{1600,1} & \cdots & v_{1600,600} & \cdots & v_{1600,700} & \cdots & v_{1600,1225} \end{bmatrix}}_{\mathrm{Ker}(\mathbf{C})} \underbrace{\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{600} \\ \vdots \\ \alpha_{700} \\ \vdots \\ \alpha_{1225} \end{bmatrix}}_{\delta\boldsymbol{\alpha}}. \tag{7.3}$$

It is neither possible nor necessary to plot all the modes, hence only a part of modes are plotted as examples. For better comparison, the following deformation modes are produced by the same perturbation magnitude for each design variable.

7.5.1 Deformation mode locate near the cut of nullspace

The deformation modes corresponding to α_1 and α_2, which with singular values just below the cut-off are depicted in Fig. 7.8. As can be seem, both modes only affect the small patches (labelled as smaller patch in Fig. 7.2) of the middle section. Theoretically, these two modes are not in the nullspace, thus they are possible to affect the continuity. However, as explained in Section 5.4.4, in numerical computing and this study, a cut-off value is used to determine the numerical nullspace which includes these two modes. Therefore, the modes near the cut-off value mainly affect the smaller patches, which are close to the interfaces between free patches thus may violate the continuity.

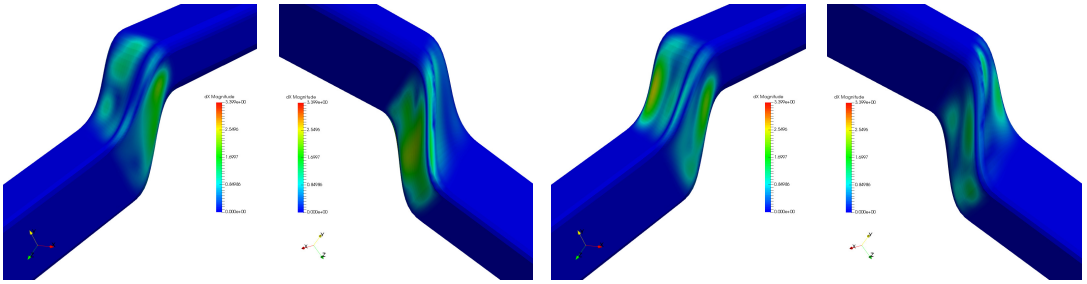


(a) α_1 (b) α_2

Figure 7.8: The deformation modes corresponding to α_1 and α_2.

7.5.2 Deformation mode locate in the middle area

The deformation modes corresponding to α_{600} and α_{700}, which locate in the middle range of $\boldsymbol{\alpha}$, are plotted in Fig. 7.9. It can be seen that these two design variables have larger influence on the shape, almost on every design surface. Firstly, these two modes are away from the cut-off value, thus the possibility of violating the continuity is smaller compared to α_1 and α_2. As can be seen, these two modes, especially the one corresponding to α_{700}, affect the smaller patches to a smaller extent than those of α_1 and α_2. Secondly, these two modes are closer to the exact nullspace, thus having larger effect on the larger surfaces (labelled as larger patch in Fig. 7.2), because the deformation inside these larger patches may not affect the continuity at patch boundaries. Combining these two factors, we have the deformation pattern shown in Fig. 7.9.

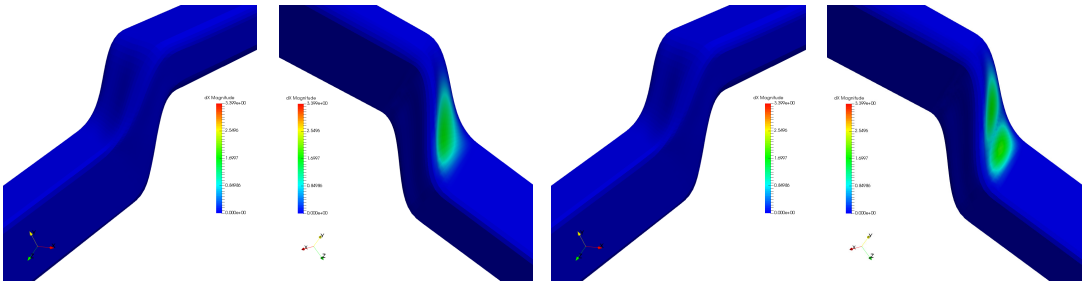


(a) α_{600} (b) α_{700}

Figure 7.9: The deformation modes corresponding to α_{600} and α_{700}.

7.5.3 Deformation mode in the exact nullspace

Two deformation modes in exact nullspace, namely those related to α_{1223} and α_{1224}, are illustrated in Fig. 7.10. The figure indicates that these two design variables mainly perturb surface parts which do not affect the continuity between moveable patches. This is because these two modes are in the exact nullspace, thus satisfying the continuity better.



(a) α_{1223} (b) $\alpha_1 224$

Figure 7.10: The deformation modes corresponding to α_{1223} and α_{1224}.

In summary, in this section the deformation modes corresponding to design variables at different parts of the design vector are plotted. These modes clearly show that different design variables affect the geometry in various manners. Generally speaking, the influence on geometric continuity between free patches decreases from the first elements of the design vector $\boldsymbol{\alpha}$ to the last elements. These deformation modes also indicated that the design space will be rather limited if exact nullspace is utilised. This is another reason why the numerical nullspace is used in practice, apart from the presence of rounding errors. Indeed, with the help of continuity recovery step presented in Section 5.5, relative large cut-off value could be used to determine the numerical nullspace thus providing

larger design space.

7.6 Results

For optimisations carried out in this chapter, the steepest descent method with Armijo line search introduced in Chapter 3 is utilised as optimiser. Optimisation results will be presented and analysed in this section.

7.6.1 Effect of cut-off value on optimisation results

Previous discussions show that the design space is a linear combination of all deformation modes, and different modes affect the geometry in different manners. Therefore, the number of deformation modes in the design space, determined by the numerical rank, will have significant effect on the optimisation results. In this section, a series of S-bend shape optimisations with different manually chosen cut-off values σ_C are carried out to investigate this effect.

Table 7.1 lists the chosen cut-off values and corresponding number of deformation modes. This table indicates that larger cut-off value leads to more deformation modes in the design space, which can also be observed in Fig. 7.7

Table 7.1: The cut-off value and corresponding number of deformation modes.

Cut-off value	Number of deformation modes
2×10^{-10}	1110
10^{-7}	1213
10^{-6}	1225
10^{-5}	1241
10^{-4}	1259

The convergence histories of the objective function corresponding to different cut-off values are given in Fig. 7.11. Note that in these optimisations, the continuity recovery steps introduced in Section 5.5 are applied. A threshold value 10^{-5} is utilised to control the recovery steps as suggested by Xu et al. [48]. Figure 7.11 illustrates clearly that both the convergence speed and the final results are affected by the cut-off value. This is achieved by changing the cut-off threshold σ_c which affects the number of deformation modes in the design space. To be more specific, when σ_C is 2×10^{-10}, the cost function

is decreased by only 20% with the slowest convergence rate, compared to those of 22.3% when using other values. This is because that fewer deformation modes are contained in the design space. Besides, it takes slightly more iterations to reduce the cost function by 22.3% when using 10^{-7} than using larger value. Finally, it can be seen that the cost function convergence histories are almost the same when the cut-off value larger equal than 10^{-6} are used.

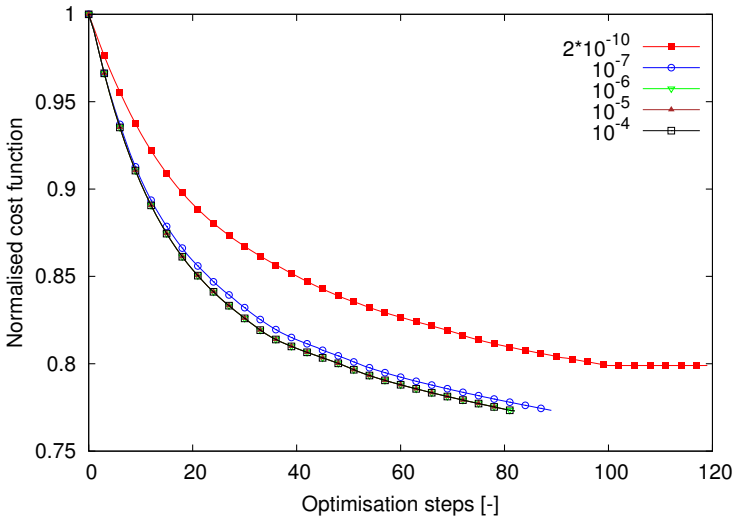
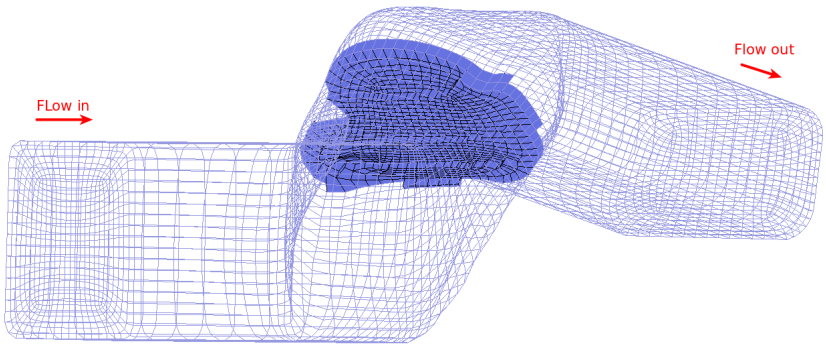


Figure 7.11: The convergence histories of the objective function when using different cut-off values.

Figure 7.12 illustrates cross section shapes of the optimised bend related to different cut-off values. It can be seen that the cross section shape is also dependent on the cut-off value σ_C. Broadly speaking, larger σ_C leads to more deformation, but very similar cross section shape is obtained if the cut-off value continues to increase after 10^{-6}. This behaviour is also observed in the cost function convergence history presented in Fig. 7.11.

By checking the cross section shape when $\sigma_C = 2 \times 10^{-10}$, one can see that the deformations are mainly on four larger surfaces, while four smaller patches do not exhibit much deformation. This is because when σ_C is very small, the numerical nullspace is closer to the theoretical nullspace, thus deformation modes mainly affect larger patches, as shown in Section 7.5.3. When the value of σ_C increases, more deformation modes affecting smaller patches (see Section 7.5.1) or both (see Section 7.5.2) are included into the design space, thus larger deformation is obtained.



(a) The cross section location

(b) Initial (c) 2×10^{-10} (d) 10^{-7}

(e) 10^{-6} (f) 10^{-5} (g) 10^{-4}

Figure 7.12: Cross section shapes of S-bend before (b) and after (c-g) optimisation.

To investigate why the final results are similar when σ_C is larger than 10^{-6}, another series of optimisations are performed without the continuity recovery steps. Figure 7.13 presents the cross section shapes with $\sigma_C = 10^{-6}, 10^{-5}, 10^{-4}$ without recovery steps, respectively. As can be seen, the optimal cross section shapes exhibit some differences with each other. However, after optimisation with continuity recovery steps, the cross sections are very similar as shown in Fig. 7.12. Therefore, it is the recovery steps that make the final results similar when σ_C is $10^{-6}, 10^{-5}$ and 10^{-6}.

(a) 10^{-6} (b) 10^{-5} (c) 10^{-4}

Figure 7.13: Cross sections of S-bend after optimisation without recovery steps.

Apart from the cost function convergence history and cross section shape, the deviation value from exact G_1 continuity after optimisation is also affected by the cut-off value, as illustrated in Table 7.2. Broadly speaking, the deviation value tends to increase along with the cut-off value σ_C. For the cases when $10^{-7}, 10^{-6}$ and 10^{-5} are used, the deviation value are successfully recovered to be smaller than the threshold with two recovery steps. However, when σ_C is 10^{-4}, the recovery steps are unable to take the deviation value back to the threshold.

Table 7.2: The G_1 deviation value after optimisation.

Cut-off value	G_1 deviation value
2×10^{-10}	8.81×10^{-5}
10^{-7}	1.15×10^{-6}
10^{-6}	1.17×10^{-6}
10^{-5}	2.27×10^{-6}
10^{-4}	1.68×10^{-5}

Based on the comparison of cost function convergence histories, cross section shapes and the G_1 deviation values, it can be seen that 10^{-6} is the best choice among all chosen cut-off values. In addition, the cut-off value is indeed a trade-off between the size of design space and the continuity constraint. One should be careful that if the value σ_C is too large, the continuity recovery steps may not be able to recover the continuity successfully.

7.6.2 Optimisation results with the effective rank

In Section 5.4.8, the *effective rank* is proposed as a pre-conditioner to determine an effective nullspace. The basic idea is to determine rank based on $\mathbf{C}\mathbf{C}^T$ instead of $\mathbf{C}$,

where $\mathbf{C}$ is the constraint matrix. Table 7.3 presents the effective rank, as well as the numerical rank corresponding to different cut-off values. It clearly shows that the effective rank is very close to the numerical rank obtained with a cut-off value of $\sigma_C = 10^{-6}$, which has been shown to give good optimisation performance. This demonstrates the feasibility of the effective rank. Therefore, the effective rank is utilised in practical optimisation of this S-bend air duct case and the corresponding optimisation results are presented in this section. It is worthwhile mentioning that by using the effective rank, the numerical space is automatically determined, one does not need to choose the value of σ_C manually thus saving a lot of time arise from trial and error.

Table 7.3: Cut-off values and corresponding numerical rank, as well as the effective rank in S-bend air duct case.

Cut-off value	Rank	Effective rank
2×10^{-10}	490	
10^{-7}	387	
10^{-6}	375	374
10^{-5}	359	
10^{-4}	341	

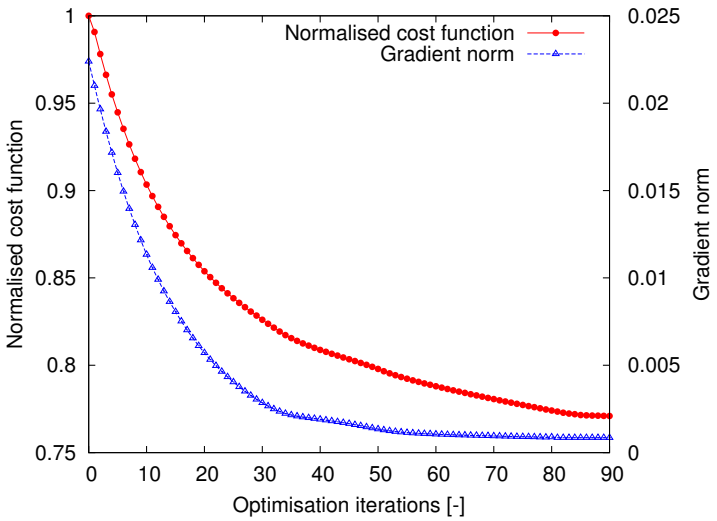


Figure 7.14: Cost function and gradient convergence history using effective rank.

The cost function and gradient convergence history during optimisation are shown in Fig. 7.14. The results verify that both the cost function and the gradient continuously reduce with the evolution of optimisation iterations. Specifically, the cost function

is reduced by about 22.9% after 90 optimisation iterations, and the gradient norm is reduced significantly as well. The surface sensitivity on the initial and optimal shape are illustrated in Fig. 7.15. As can be observed, the sensitivity is reduced significantly almost everywhere, except at interfaces of moveable patches with the fixed inlet and outlet.

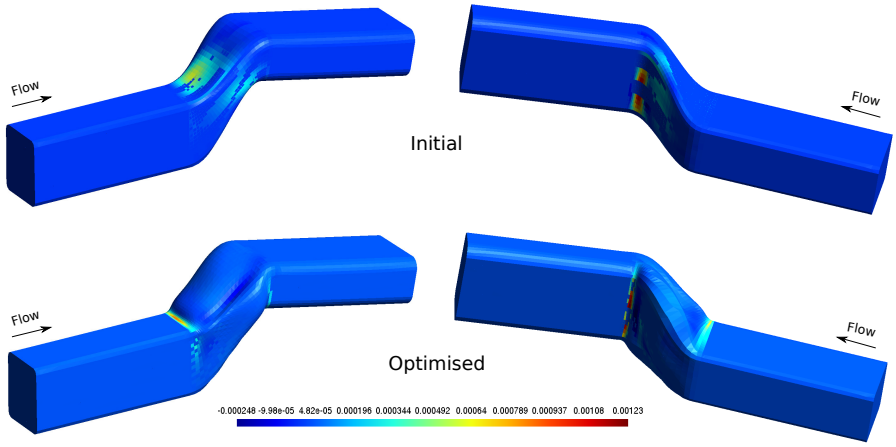


Figure 7.15: Surface sensitivity on the initial and optimised shape.

The geometry before and after optimisation are illustrated in Fig. 7.16, where a large deformation can be observed on the design surfaces, especially the squeezed-in sides. To have a better understanding on the result we plot the contour plots of velocity magnitude at different cross sections, as shown in Fig. 7.17 and 7.18. The streamlines of the flow are given in Fig. 7.19. It can be seen that the initial flow field has a strong secondary flow, and the cross-sectional cuts are highly non-uniform in terms of velocity magnitude. This kind of secondary flow in bent ducts is known as Dean vortices [252], as depicted in Fig. 7.20, which will increase the pressure loss. Figure 7.18 also indicates that there is a large flow separation at the bottom of the outlet leg, which will additionally gives rise to pressure drop.

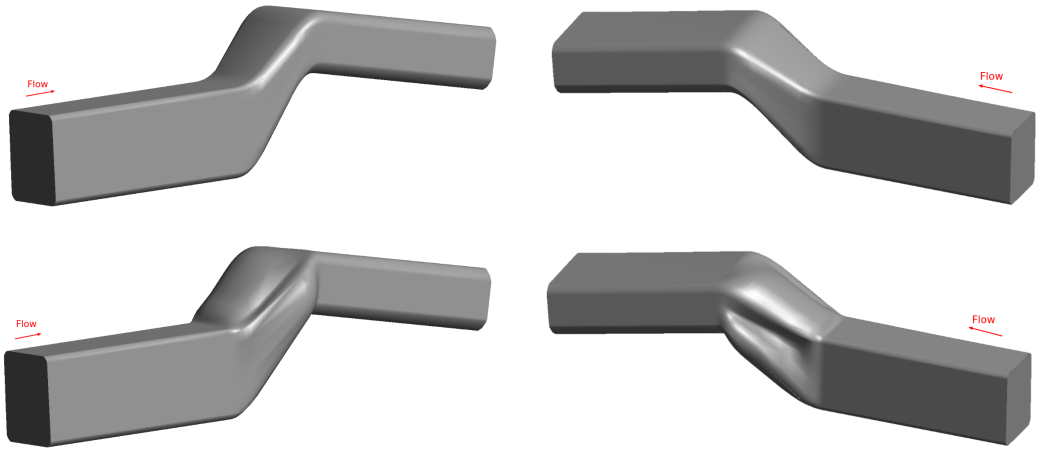


Figure 7.16: Comparison between the initial (top) and optimised (bottom) shape.

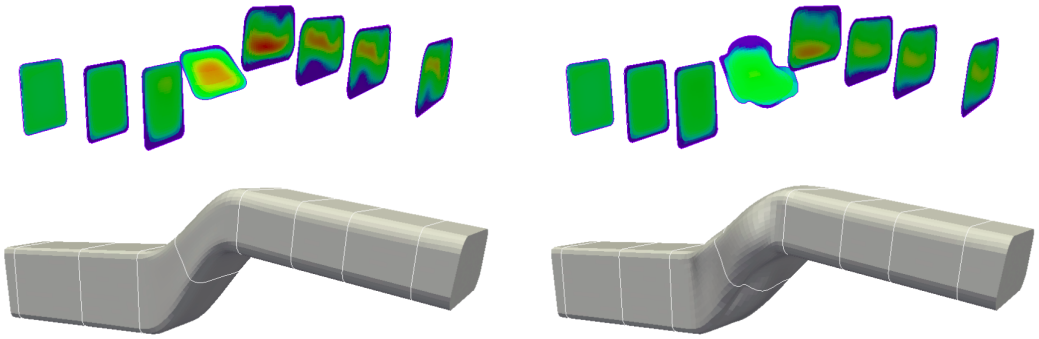


Figure 7.17: Contour plots of velocity magnitude for the initial (left) and optimised (right) S-bend duct at different cross sections along the flow direction.

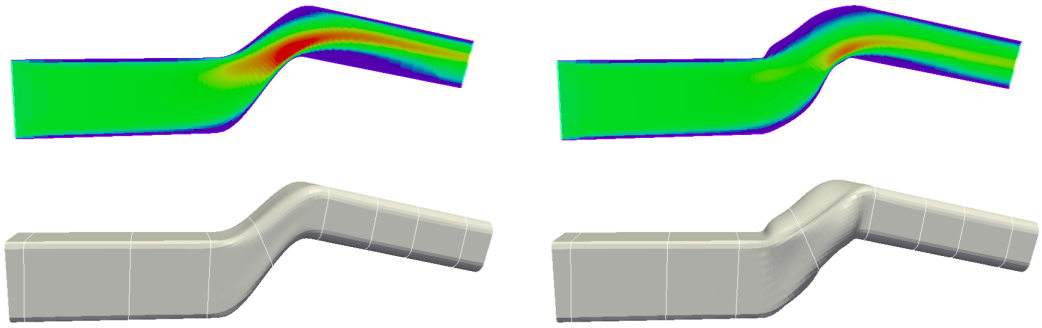


Figure 7.18: Contour plots of velocity magnitude for the initial (left) and optimised (right) S-bend duct at a cross section parallel to the inlet flow direction.

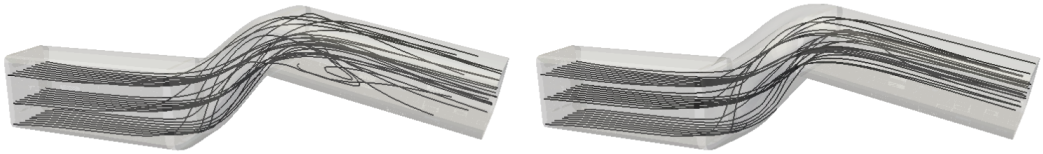


Figure 7.19: The comparison of streamlines before (left) and after (right) the optimisation

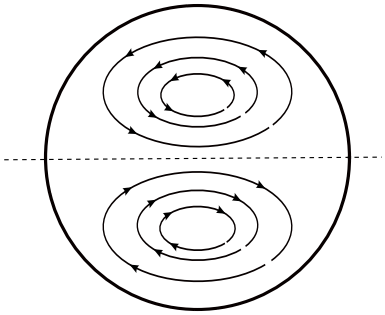
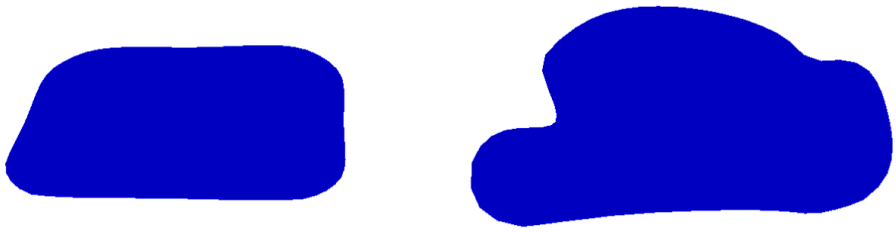


Figure 7.20: Schematic of a part of Dean vortices in curved pipes.

Physical explanation of the results

Now that Dean vortices are one of the main reason for pressure drop, then it is important to reduce them. One of the measures to reduce the Dean vortices, is using strake-like shape to suppress the formation of vortices, as demonstrated by Xu et al. [48]. The hollowed sides in the optimal shape in this study actually resembles the strake-like shape, as illustrated in Fig. 7.16. A more clear illustration of the cross cut of the cranked part, at the same location as in Fig. 7.12, is shown in Fig. 7.21. It can be

seen clearly that the side surfaces are pushed inwards, such that the suppression of flow separation is achieved.



(a) Initial (b) Optimised

Figure 7.21: The cross section before (left) and after (right) the optimisation.

Thanks to suppressed Dean vortices, the flow field exhibits more uniform cross-sectional cuts after optimisation, as can be seen from the right of Figs. 7.17 and 7.18. The significant reduction in secondary flow motion is also visible in Fig. 7.19, showing that the streamlines are much more smooth in the deformed shape. In addition, the velocity magnitude contour and streamlines illustrate that the flow separation is significantly reduced in the optimised shape.

The other reason of reduced separation in the optimised shape is the enlargement in the bend. To be more specific, the cross-section area firstly increase and then contract along the flow direction, mitigating the adverse pressure gradient hence can reduce the flow separation. As a result, lower total pressure loss is achieved after optimisation, which is confirmed by the pressure distributions along the bend shown in Fig. 7.22. Lower flow velocity in the bend reduces the required pressure gradient to turn the flow, thus the pressure at the inlet is reduced after optimisation.

Comparison with results in the literature

This S-bend air duct testcase was provided to the FlowHead research project by Volkswagen AG. It served as a testcase, and was frequently used in that project and follow-on projects (e.g. AboutFlow) for the purpose of solver verification [169, 253, 254] and shape optimisation [48, 167, 192, 255].

Xu et al. [48] performed CAD-based shape optimisation for the S-bend duct. In that work, the GPDE flow solver and an older version of NSPCC CAD kernel were utilised. The comparison of optimisation results from the present study and [48] is illustrated in Fig. 7.23. As can be seen, although the results are not exactly the same, very similar deformation patten are achieved. To be more specific, strake-like shape is formed

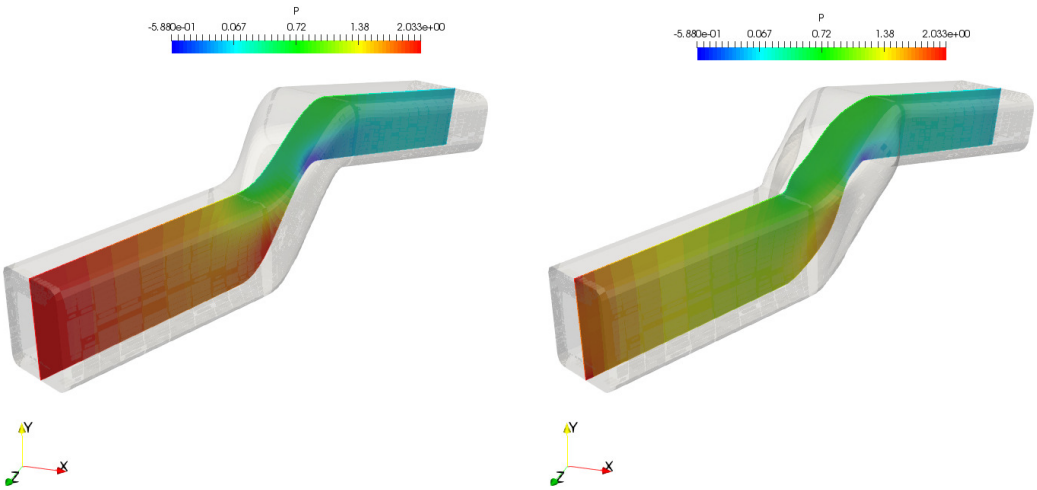


Figure 7.22: Pressure field distribution before (left) and after (right) optimisation.

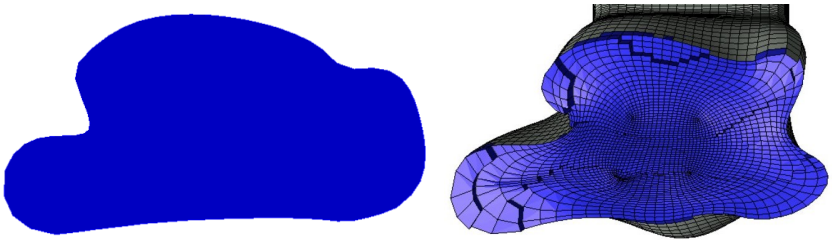


Figure 7.23: The cross section after optimisation in the present study (left) and that presented in [48] (right).

and the bend is enlarged compared to initial shape. The differences between results in the present work and those published in [48] maybe due to different implementation of optimiser (e.g. step size), different number of modes in the design space, etc.

Although this S-bend air duct used is not a standard benchmark case which is widely used in academia, it is still a suitable test case in the present work for following reasons. Firstly, validation has been performed by comparing results with those provided by OpenFOAM (Section 4.2.1). Second, published results [48] are available for comparison. Thirdly, total gradient validation presented in Section 7.4 indicates that the flow sensitivities from adjoint solver, the geometric sensitivities from CAD kernel, and the sensitivity assembly are correct for this case. In addition, the geometry is relative simple yet it comes from the industry, thus can show the practical usage of the developed method in this study. Therefore, in the present work the S-bend air duct is also chosen as a test case.

There are also some other studies in the literature which optimise the shape of S-

bend duct, although the geometries are not exactly the same. For example, Othmer and Grahs [8] performed both the shape and topology optimisation for the S-bend duct, aiming at exploring the benefits of CFD optimisation in the car development process. The interesting readers could refer to those research.

7.6.3 Continuity recovery results

In this S-bend shape optimisation, G_1 continuity constraint is imposed among moveable surfaces. As discussed in Section 5.5, both the non-linearity of G_1 continuity and the inaccuracy of nullspace due to numerical computing will violate the G_1 continuity. Consequently, continuity recovery steps are required to make sure G_1 is better satisfied. As presented in Section 7.6.1, continuity recovery steps are control by the threshold 10^{-5}.

Figure 7.24 shows the deviation value from exact G_1 with and without recovery steps. It is indicated that the G_1 deviation value are negligible after applying recovery steps, compared to those without recovery steps. Specifically, in the present optimisation case, two recovery steps can bring the G_1 deviation value below the chosen threshold.

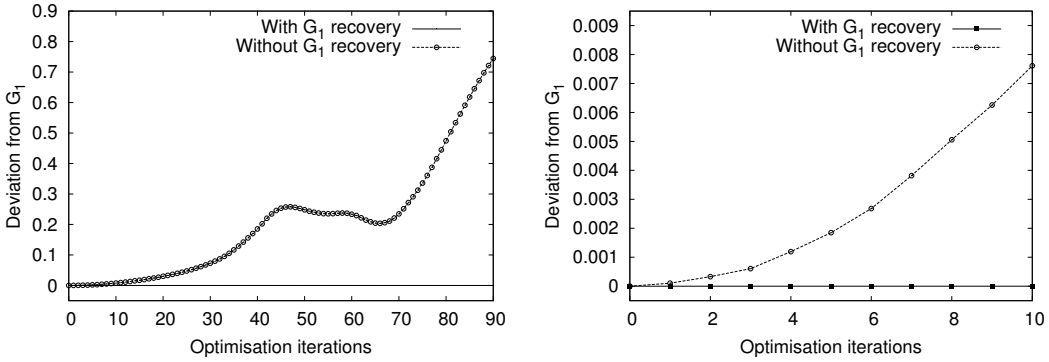
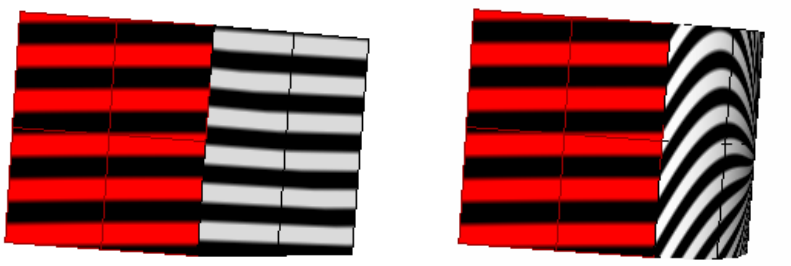


Figure 7.24: Deviation from exact G_1 continuity with and without recovery steps. Left: during 90 iterations. Right: details in the first 10 iterations.

The Zebra continuity analysis is a tool provided by most commercial CAD software to inspect the continuity between surfaces[2]. In the seam where two surfaces meet, if the Zebra stripes have kinks or jump sideways, this indicates G_0 continuity. G_1 continuity means that the stripes line up but turn sharply at the connection.

[2]http://docs.mcneel.com/rhino/5/help/en-us/commands/zebra.htm



(a) G_0 continuity (b) G_1 continuity

Figure 7.25: Example of Zebra stripes of G_0 and G_1 continuity[3].

The Zebra analysis is performed for the optimised S-bend geometry, as illustrated in Fig. 7.26, which indicates that the G_1 continuity among free patches are satisfied.

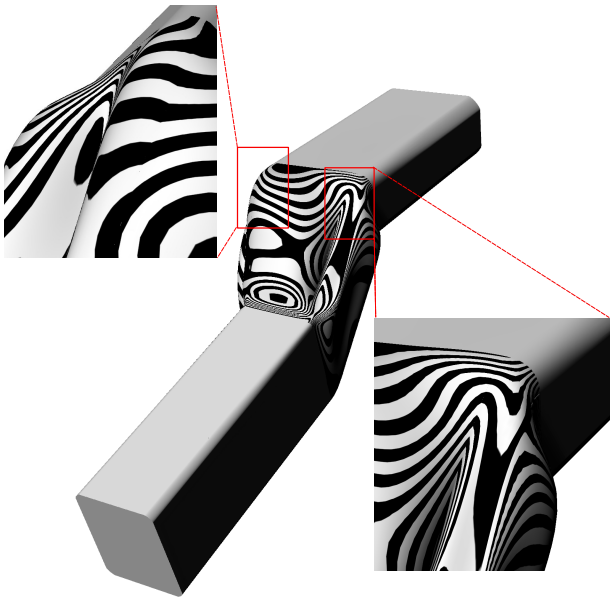


Figure 7.26: Zebra continuity analysis of the optimised shape.

7.6.4 Effect of mesh density on optimisation results

The accuracy of CFD solutions is dependent on mesh density. It would be helpful to investigate how the mesh density affects results of this S-bend air duct shape optimisation. To this end, optimisation with a denser mesh (120K) has been carried out. All the setting parameters and solver are the same as previous case with 44K mesh presented in Section 7.6.2.

The comparison of initial flow field is presented in Fig. 7.27. As can be seen, both cases exhibit similar flow pattern and have large flow separation although the mesh densities are quite different.

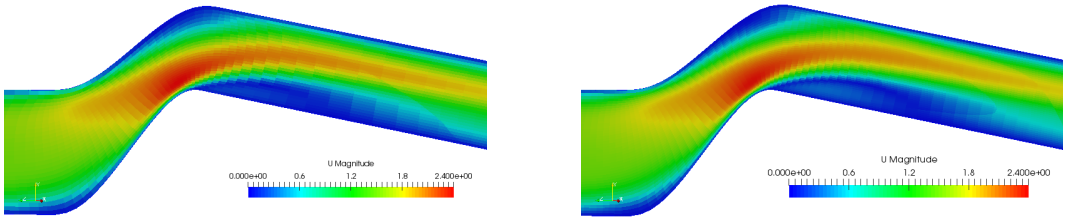


Figure 7.27: Comparison of initial flow field between 44K (left) and 120K (right) mesh.

The comparison of cost function convergence histories and gradient norm between 44K and 120K mesh are shown in Fig. 7.28, which indicates that the results are slight better when finer mesh is used. Specifically, by using the 120K mesh the optimisation converges in fewer iterations and about 1.5% more pressure drop reduction is achieved.

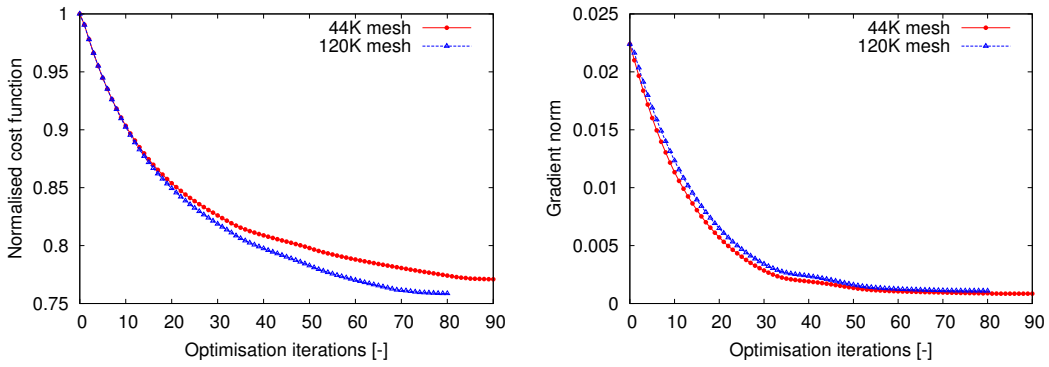


Figure 7.28: Comparison of convergence history between 44K and 120K mesh. Left: cost function. Right: gradient norm.

The contour of velocity magnitude for both cases are plotted in Fig. 7.29 and Fig. 7.30, respectively. It can be seen that, even though the mesh density is increased to almost three times of the previous case, the velocity magnitude show very similar pattern between them. The comparison of optimal shape is also given in Fig. 7.29, where similar deformation can be observed.

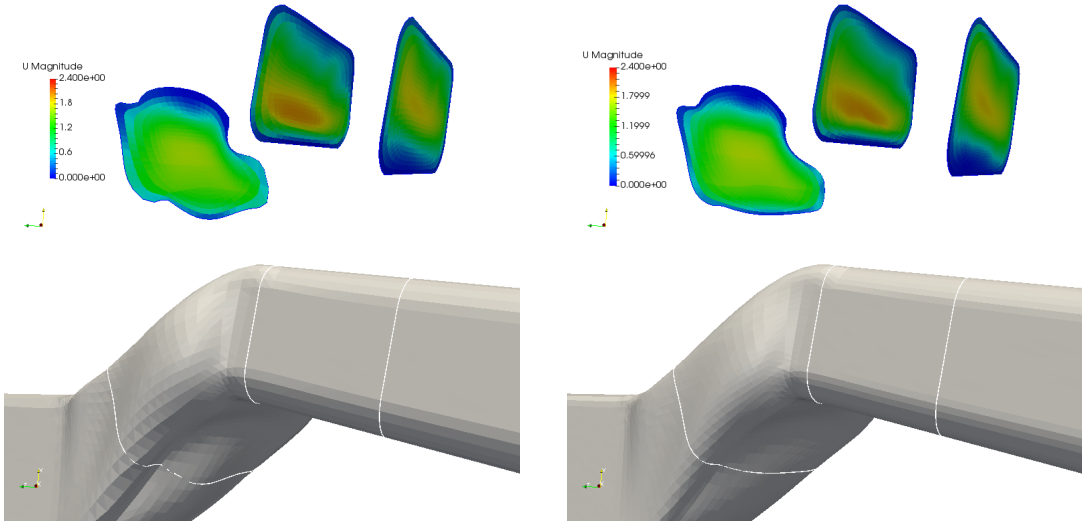


Figure 7.29: Contour plots of velocity magnitude for the 44K (up left) and 120K (up right) case at cross sections along the flow direction, and corresponding shape (bottom).

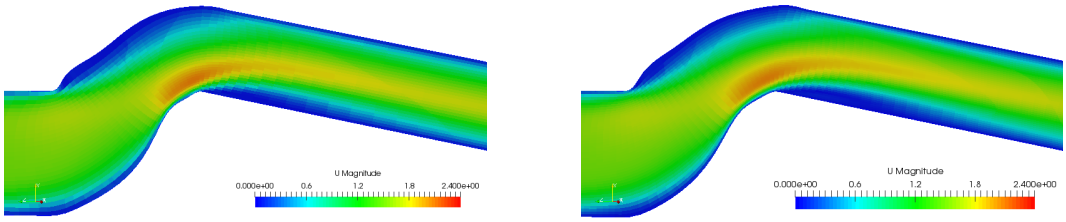


Figure 7.30: Contour plots of velocity magnitude for the 44K (left) and 120K (right) S-bend duct at a cross section parallel to the inlet flow direction.

The optimised cross section shape at the location in Fig. 7.12 are compared in Fig. 7.31. Again, the deformation in both cases show similar behaviour, namely the side surfaces are hollowed in to form the strake-like shape such that suppress the Dean vortices.

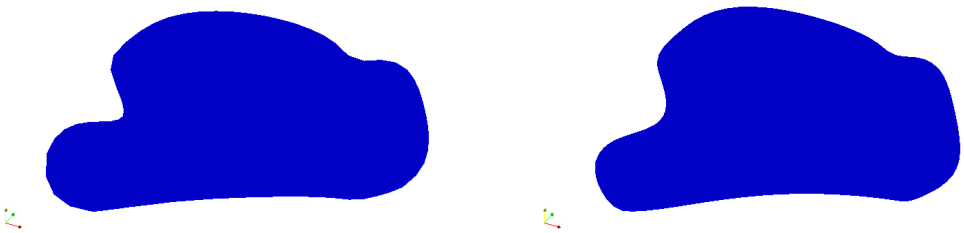


Figure 7.31: Comparison of optimised cross section: 44K mesh (left) and 120K mesh (right).

Based on these comparisons, one can see that the 120K mesh does not provide radical improvements compared to the 44K mesh, but offers small enhancements. In other words, the mesh density does not affect the results of this S-bend air duct optimisation case significantly. The possible reason is because the flow in this S-bend air-duct case is laminar (the Reynolds number is 300) and not very complicated. Therefore, the 44K mesh is enough for capturing the main flow pattern and adequate for shape optimisation of this bend.

7.6.5 Effect of scaling on optimisation performance

In previous optimisations presented in this chapter, the weights of control points are frozen. In this section, we try to add the weights into the design space as well. As mentioned in Section 5.6, the scaling problem arises when both position and weight of control points are free to change during the optimisation, which may deteriorate the performance of optimiser. The S-bend optimisation case has been performed with free control point position and weight to demonstrate the effects of scaling on optimiser performance.

The diagonal scaling strategy presented in Section 5.6 is applied to homogeneous control points to make all components have similar order of magnitude. The objective function convergence history is plotted in Fig. 7.32 together with that of non-scaling case. As can be seen, the cost function decreases much faster if scaling is applied, and around more than three times reduction has been obtained. These results demonstrate that the optimisation performance is truly affected by scaling, if both the position and weights are changeable. The comparison also gives a guidance to the user of CAD-based optimisation framework that, if design variables with different scales are used, such as position and weight of control points, or angle of attack and planform variable in airfoil optimisation, a proper scaling is essential for good optimiser performance.

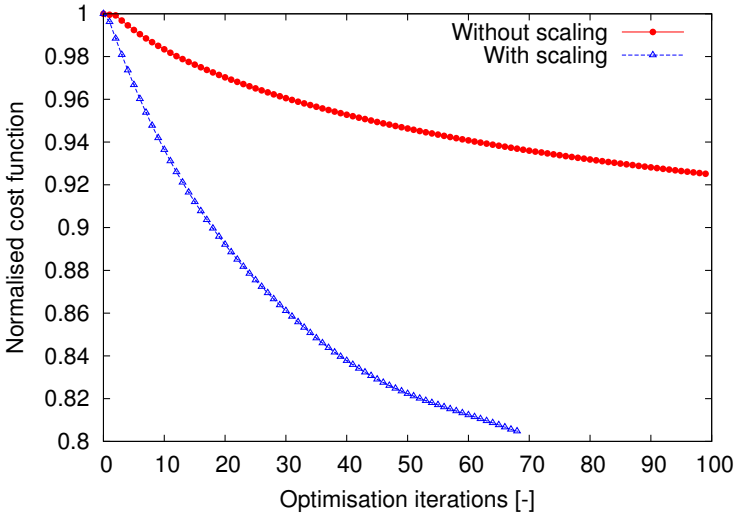


Figure 7.32: Comparison of cost function convergence history with and without scaling.

7.7 Summary

In this chapter, the NSPCC approach is applied to the optimisation of an S-bend air duct from automotive industry. The feasibility of the NURBS-based optimisation framework, coupling NSPCC and an incompressible flow solver as well as a discrete adjoint solver, has been demonstrated.

Deformation modes in the numerical nullspace have been plotted to show their impacts on geometry. The results show that deformation modes in different areas of the numerical nullspace impact the geometry and geometric continuity in different manner. Based on this fact, the effect of cut-off value, that is the number of deformation modes included in the design space, on the optimisation results are investigated. It has been shown that the cut-off affects the optimisation performance significantly. Therefore, it is crucial to choose a proper cut-off value.

This chapter also presents the optimisation results using the effective rank. After 90 optimisation iterations, around 22.9% total pressure loss reduction is achieved while keeping the smoothness around the surface interfaces. This indicates that the effective rank is a proper pre-conditioner to determine a numerical rank leading to effective design space. What's more, by using the effective rank, the manual effort to choose a cut-off value σ_C which has good performance by trial and error are avoided, hence the optimisation loop becomes more automatic and easier to use.

In addition, this chapter has discussed the influence of the mesh density on the final results. For this laminar S-bend case, the 120K mesh gives slightly better results compared to that of the 44K mesh, but both cases exhibit similar deformation trend of the bend.

Finally, the impact of the scaling of control points when weights are added into the design space has also been investigated.

Chapter 8

Shape optimisation of a U-bend cooling channel

The NSPCC approach is applied to shape optimisation of a U-bend cooling channel used in turbo-machine industry, aiming at reducing the pressure drop in the bend. The U-bend geometry contains a circular part which cannot be described with B-splines exactly. As a consequence, the B-spline version NSPCC [48] cannot handle this case. Therefore, the U-bend constitutes a suitable test case to show the effectiveness of the extended NSPCC developed in this research. Other new capabilities, such as coupling with an advanced optimiser and imposing different levels of continuity constraints in one geometry, will also be demonstrated in this chapter.

8.1 Introduction

The turbines used in gas turbine engines operate at extremely high temperatures, which common materials cannot withstand, to achieve high thermal efficiencies. Therefore, effective cooling is essential. This is typically achieved by utilising an internal cooling channel. The U-bend shapes which connect consecutive passages have been widely used (see Fig. 8.1) and have attracted much attention in recent years [25–27, 256]. For example, the U-bend cooling channel was used as a benchmark test case in the 2016 International Conference on Numerical Optimisation Methods for Engineering Design (NOED 2016)[1].

However, a salient feature of the cooling fluid flowing along the bends is that it has pressure loss. The pressure drop increases the pressure required at the inlet of the cooling

[1] http://aboutflow.sems.qmul.ac.uk/events/munich2016/

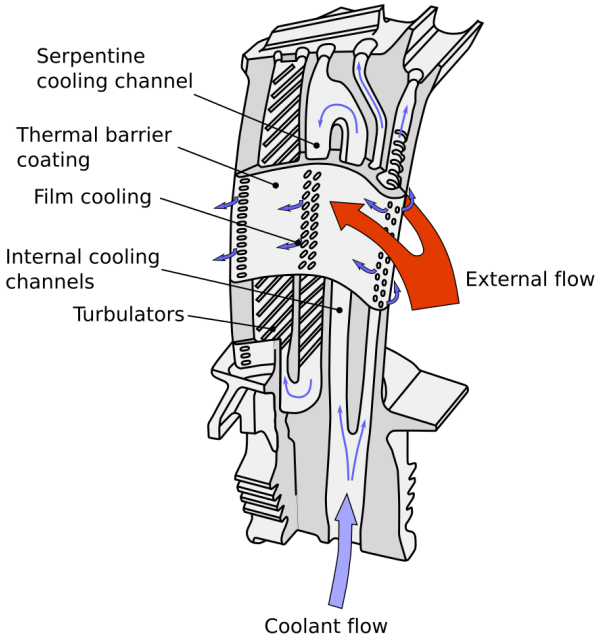


Figure 8.1: Internal cooling channel in the turbine [257].

system, and could be problematic for the cooling system. Therefore, it is necessary to reduce the pressure loss.

In the past, the reduced pressure loss was obtained relying on manual design, which was laborious and time-consuming. Recently, the design process of U-bend have become more automatic with the help of numerical optimisation [25–27, 45, 204, 256, 258]. Some of these works are based on gradient-free optimisation methods. For instance, Verstraete et al. [26] optimised a 2-D U-bend using a metamodel-assisted evolutionary algorithm to reduce the pressure loss, and the number of DoF of the geometry parametrisation in this study was 26. Experimental validation results for the optimal shape were then presented by Coletti et al. [27]. In a follow-on study, Verstraete et al. [204] performed a multi-objective optimisation for a 2-D U-bend, aiming at minimising the pressure loss and maximising the heat transfer ability simultaneously. The parametrisation used in [26] with 26 parameters was utilised again, and a gradient-free method was used as optimiser. Namgoong et al. [256] optimised the inner wall of both 2-D and 3-D U-bend configurations, with 8 and 24 design variables, respectively. Both optimisation results obtained with genetic algorithm and experimental validation results were reported in this study. Although pressure loss reduction were achieved by utilising gradient-free methods in these studies, only a small number of design variables were used due to the computational cost. Therefore, the design space may have been too limited.

In order to use a larger number of design variables, gradient-based methods are also

employed recently to explore better U-bend shape design. For example, Willeke et al. [25] carried out a 2-D U-bend optimisation using a continuous adjoint method to provide computationally inexpensive gradients. The optimiser was the steepest descent algorithm. Two parametrisations were utilised in this work, namely the node-based method (see Section 2.3.1), and Bézier curve-based parametrisation. For the node-based case, there was a large number of design variables which was only feasible with gradient-based method in CFD shape optimisation. It was shown in this study that the gradient-based method produced a slightly larger pressure loss reduction. Akin et al. [258] optimised a quasi-3D U-bend described with B-spline curves. Since the gradients were computed with FD in this study which was expensive, thus still only 21 and 25 design variables were used. Banovic et al. [100] performed a 3-D U-bend shape optimisation by using the differentiated OCCT CAD kernel coupled with a discrete adjoint solver. The parametrisation was based on a cross-sectional design approach, and 96 design parameters were available. Although the pressure loss was reduced by 18%, it seems that there was still very large flow separation in the optimal bend. A recent study on the U-bend shape optimisation was performed by Verstraete et al. [46], where tri-variate B-splines was employed to describe the U-bend volume with 540 DoF. A discrete adjoint solver was employed to compute gradients efficiently and the steepest descent method was used as optimiser.

In these studies, Bézier or B-spline curves were utilised to approximate the circular part in the U-bend geometry. One exception is the study performed by Tsiakas et al. [259], where the U-bend geometry is embedded inside a volume based on volumetric NURBS, which is actually a FFD method (see Section 2.3.1). However, the paper did not discuss how to maintain geometric continuity among moveable patches, which is quite important in shape optimisation. The paper also did not perturb weights of control points during optimisation. Another problem of the existing gradient-based U-bend shape optimisations is that, to the best of the author's knowledge, only the steepest-descent method has been used as the optimiser. The quasi-Newton or the Newton methods have not been applied yet.

In this chapter, the results of 3-D gradient-based shape optimisations for a U-bend cooling channel aiming at reducing the pressure drop, are presented. The geometric continuity constraints between adjacent patches are also handled.

The CAD-based shape optimisation framework presented in Fig. 1.3 is utilised, including the NSPCC approach to handle geometric continuity constraints and STAMPS to compute the flow sensitivities efficiently. The analytic NURBS description of U-bend geometry is remained in the design loop and could be written out to a STEP file in each design iteration. In addition, the BFGS method is also utilised as the optimiser apart

from the steepest descent approach, trying to provide better optimisation results.

The geometry and parametrisation of the U-bend are presented in Section 8.2, followed by details on solver and setting up parameters in Section 8.3. The initial flow field is presented and analysed in Section 8.4. Then the optimisation results are given in Section 8.5.

8.2 Parametrisation

8.2.1 Geometry of the U-bend cooling channel

The U-bend cooling channel used in this work comes from the Von Karman Institute (VKI) [25–27] and is one of the benchmark test cases in NOED 2016 conference. The geometry is created by the author using NURBS surfaces according to the case description provided by NOED 2016 committee. The geometry and main dimensions of this U-bend are shown in Fig. 8.2 and Fig. 8.3, respectively. Basically, the baseline geometry is comprised by a circular bend part with inlet and outlet legs. The inlet leg is long enough to guarantee a fully developed flow at the entrance of the circular bend. The hydraulic diameter is $D_h = 0.075m$, and the rectangular cross section has aspect-ratio 1.

B-splines cannot describe circular and conic curves/surfaces precisely, thus are not able to describe the circular part in the U-bend exactly. In this work, NURBS surfaces are utilised to parametrise this U-bend geometry. In this way, no approximation is need to represent the geometry, and NURBS provide weights of control points as additional design freedoms. The U-bend geometry consists of 22 NURBS surfaces in total. The bend part, which is composed by 8 rectangular and 4 circular surfaces (shown in red in Fig. 8.3), is deformable. Each of the rectangular surfaces has 24 (6×4) control points, and each of the circular surfaces has 16 (4×4) control points. As a result, there are in total 256 control points for the bend part. Both inlet and outlet legs are fixed. The deformable surfaces and corresponding control points are also shown in Fig. 8.2.

8.2.2 Geometric continuity constraints

For this U-bend case, geometric continuity constraints are required among patches. There are 8 common edges between moveable and fixed patches, and 20 common edges between the movable patches themselves. For the former case, G_1 continuity is required. Normally,

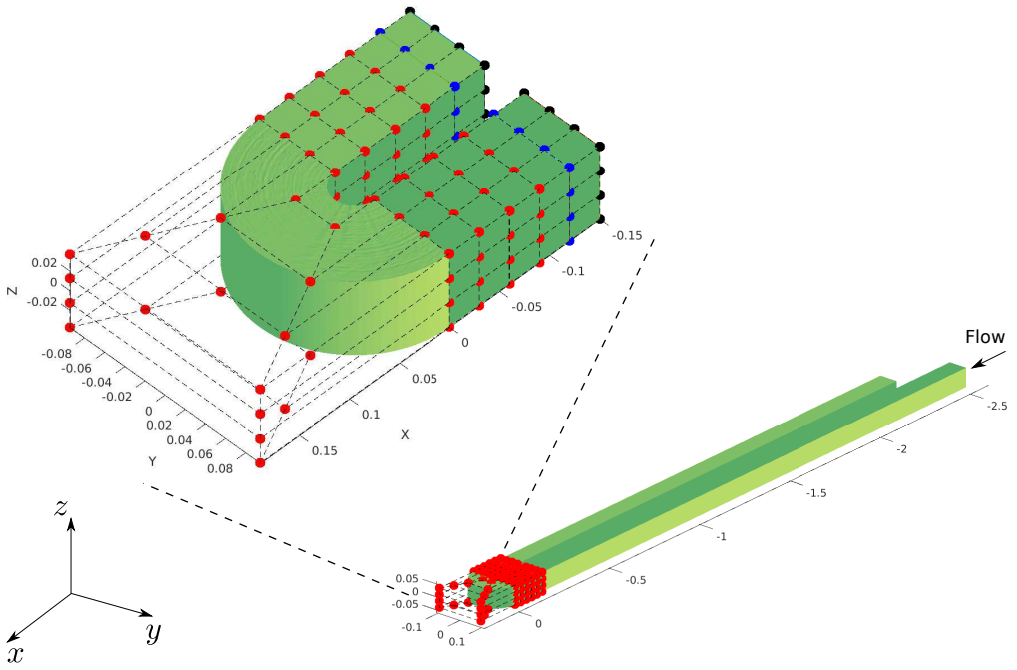


Figure 8.2: 3-D view of the U-bend cooling channel and control points of design surfaces.

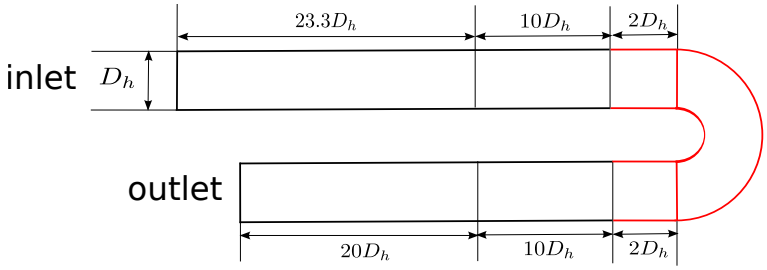


Figure 8.3: Main dimensions of the U-Bend.

this is achieved by locking the first two columns of control points of the bend part as shown in Fig. 8.4. Actually, there is no need to totally freeze the second column control points, because the G_1 continuity will be maintained as long as these control points are kept on the initial planes [51]. For example, the second column control points of the top surface should stay on the top surface. Therefore, in this study G_1 continuity is imposed by fixing only the first column of control points (shown in black in Fig. 8.2) and restricting the second column of control points (shown in blue in Fig. 8.2) to stay on their initial planes. By doing this, we can have more degrees of freedom for the geometry. As a consequence, 192 control points can move in three directions and 32 control points can move in two directions. The weights of these control points can also change.

The continuity among free patches is imposed by using the test point approach presented in Section 5.4.9. Firstly, G_0 is imposed on all 20 common edges to avoid gaps. Apart from this, G_1 continuity is imposed on the edges between rectangular and circular NURBS patches, as shown in Fig. 8.5. 12 pairs of test points are imposed if weights of control points are fixed, and 18 pairs of test points are used if weights are also free, based on the estimate proposed in Section 5.4.9.

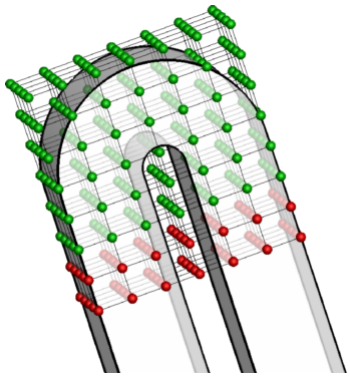


Figure 8.4: Control points marked in red remain fixed during the optimisation to maintain the G_1 continuity [259].

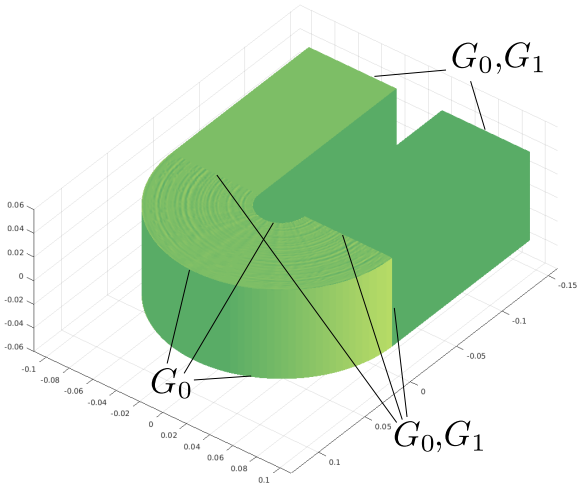


Figure 8.5: Continuity constraints imposed on different edges of U-bend.

The effective rank proposed in Section 5.4.8 is employed in the SVD to determine the size of nullspace and ultimately the number of design variables.

8.3 Solver and parameters

The design objective is to reduce the total pressure loss in the bend, which is defined as:

$$J = \frac{\int_{inlet} p_{total}(\mathbf{u}\cdot\mathbf{n})dS - \int_{outlet} p_{total}(\mathbf{u}\cdot\mathbf{n})dS}{\int_{inlet}(\mathbf{u}\cdot\mathbf{n})dS}, \tag{8.1}$$

where p_{total} is the total pressure, $\mathbf{u}$ is the velocity vector, $\mathbf{n}$ is the surface normal direction and S is the cross section area.

Main parameters in this case are as following:

- Reynolds number: 15000
- Mach number: 0.1
- Mesh density: 167K

The in-house solver STAMPS introduced in Sections 4.2.2 and 4.3.3 is employed to solve the flow equations and provide the sensitivity of cost function with respect to mesh node coordinates. The U-bend is a rather challenging case for the CFD solver, because convergence is affected by the wave reflection with long travel time between the inlet and outlet, which are needed to form fully developed flow. Although STAMPS is a compressible solver, in this case the Mach number is set to 0.1, such that an incompressible assumption can be used to model the flow. STAMPS is capbable of simulating the flow in this U-bend cooling channel, and the feasiblity of using STAMPS to simulate flow in U-bend have been demonstrated by several studies [46, 100, 260].

The Reynolds number is around 15000, based on the hydraulic diameter. Therefore, the flow is turbulent in this case. The turbulence closure problem is realized with the Spalart-Allmaras (SA) turbulence model which is implemented in STAMPS. Note that at the moment, only the SA turbulence model has been implemented in STAMPS. Although the SA model is more suitable for external flow, it still gives insight to the flow thus is also used by Verstraete et al. [46]. In this study, no heat transfer is considered.

A regular hexahedral mesh with 167K nodes and 177K cells is used, as shown in Fig. 8.6, where the surfaces to be optimised are shown in red. Since this is a turbulent case, to capture flow features in the boundary layers region better, a boundary layer refinement suitable for a low Reynolds turbulence model is performed, which is required to predict the secondary flow structures. The mesh has an average y^+ value of 1.

Jesudasan et al. [260] performed a preliminary grid convergence study using three

different mesh levels, and stated that the objective function value between medium (167K) and fine (300K) mesh is 0.3%, which is minor. In the present work, the mesh with 167K nodes is used.

At each optimisation iteration, the surface mesh is recalculated on the updated CAD geometry given by the NSPCC kernel, then surface mesh deformation will be propagated to deform the volume mesh. The inverse distance weighting method [177] implemented in STAMPS is utilised to deform the volume mesh.

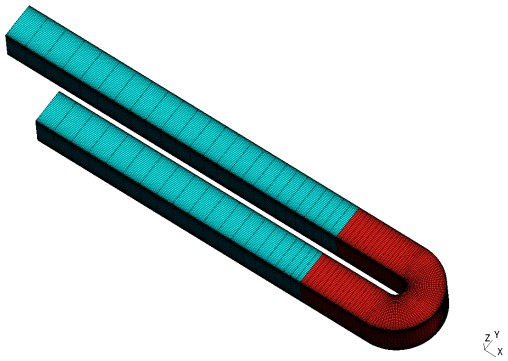


Figure 8.6: The 167K hex mesh for U-bend (design surfaces in red).

8.4 Initial flow field

The velocity magnitude of the baseline geometry at the middle height plane is illustrated in Fig. 8.7. As can be seen, there is a large flow separation area after the turn and in the downstream leg. The separation starts just before the middle of the turn. This is due to the strong curvature of the turn, thus in the inlet leg the flow accelerates quickly near the inner wall while decelerates near the outer wall. The acceleration near the inner wall leads to a low-pressure zone, which results in a strong adverse pressure gradient downstream of the flow. Figure 8.8 shows the velocity magnitude at three different cross sections. The velocity magnitude in the outlet leg shows that the flow separation reduces the effective cross-section area and accelerates the flow, which needs to be subsequently diffused when the flow fills the downstream channel. The diffusion process results in a loss of total pressure. The mid-bend contour shows that a pair of contra-rotating vortices is formed (Dean vortices), due to the imbalance between centrifugal forces and pressure gradient over the height of the channel because of the slower flow near the endwalls. This also contributes to the pressure loss.

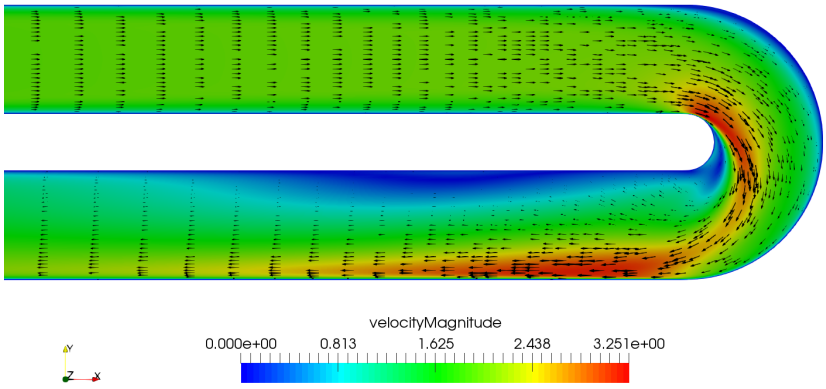


Figure 8.7: Initial velocity magnitude at the middle height plane.

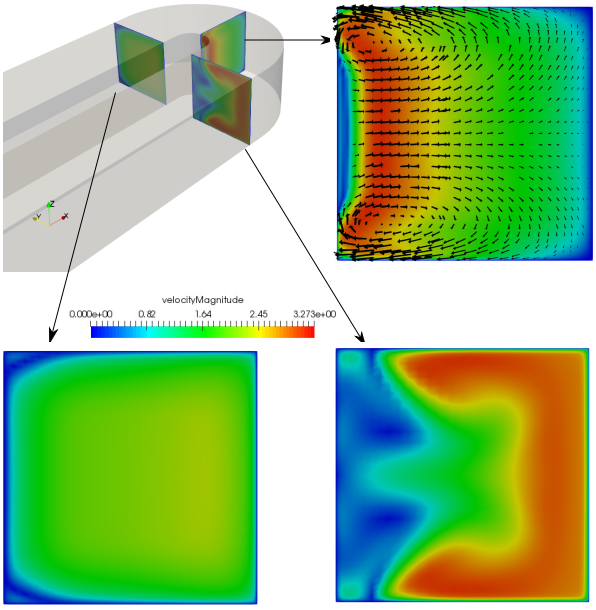


Figure 8.8: Initial velocity magnitude at different locations.

Comparison with experimental results

Coletti et al. [27] from the von Karman Institute performed experimental study on the U-bend case. In that study, a magnified model of the U-bend that reproduces the geometrical and aerodynamic features of the numerical model was built, and the particle image velocimetry (PIV) technique was utilised to measure the velocity. The Reynolds number was set to 40,000, which was the same as in [26]. Note that in [26], the same

geometry as in the present study was used. The experimental set up is shown in Fig. 8.9.

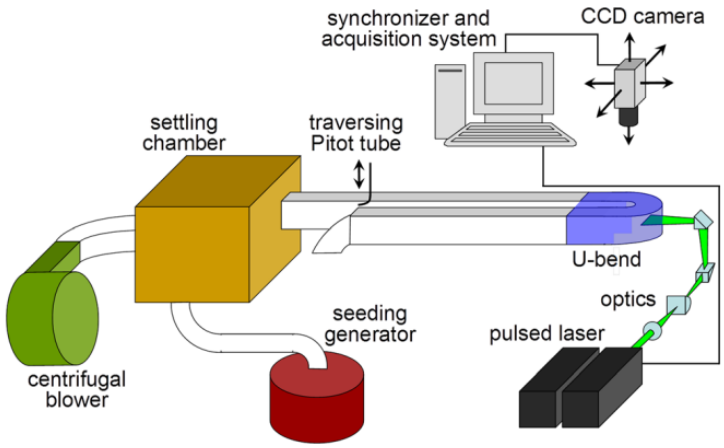


Figure 8.9: Schematic representation of the experimental set up [27].

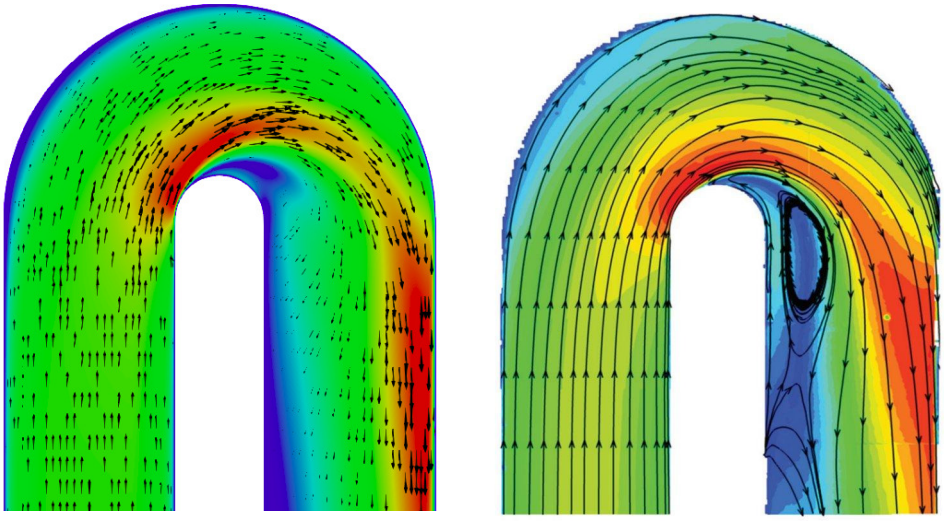


Figure 8.10: Comparison of velocity magnitude at the middle height plane of initial geometry between STAMPS results (left) and experimental results (right).

Fig. 8.10 presents the comparison of velocity magnitude at the middle height plane for the initial geometry. As can be seen, although the Reynolds number are different, the solutions given by STAMPS flow solver exhibit very similar features as shown in the experimental data. In other words, the STAMPS flow solver captured the main flow features, namely the acceleration along the inner wall, the deceleration near the outer

wall, and the large flow separation area after the turn. This qualitative comparison with experimental data indicates that STAMPS is capable of providing reasonable solutions for this turbulent U-bend case. The little differences between simulation and experimental results may come from the different Reynolds number. Besides, the SA turbulence may not capture this internal flow very well.

The aim of the optimisation is to reduce the total pressure loss, it is expected that after optimisation the flow separation will be suppressed.

8.5 Results

8.5.1 Optimisation results with the steepest descent method

As mentioned in Section 5.4.7, if non-linear constraint is imposed and the constraint matrix is updated in each design iteration, the size of nullspace may change. As a consequence, the number of design variables may not be a constant during the optimisation run. Therefore, as a first case the steepest descent method with Armijo line search is chosen as optimiser in this U-bend optimisation. The design variable is updated as:

$$\boldsymbol{\alpha}_k = -s_k \cdot \mathbf{g}_k \tag{8.2}$$

where $\boldsymbol{\alpha}_k$ is the design variable vector in the k-th optimisation iteration, s_k is the step size, $\mathbf{g}_k$ is the gradient of cost function in iteration k. Note that in this case the weights of control points are frozen, so the total number of DoF is 640 ($192 \times 3 + 32 \times 2$).

Figure. 8.11 presents the convergence history of cost function. As can be seen, the total pressure loss is reduced by around 21.5% after 45 optimisation iterations.

The velocity magnitude at the middle height plane of the optimised geometry is given in Fig. 8.12. As can be seen, the flow separation presented in the baseline geometry has been reduced significantly after optimisation.

Physical expanation of the results

As can be seen from Fig. 8.12, the thickness of the divider wall, namely the space in between the forward and return channel, of the bend is increased. Besides, the radius of the inner wall in both the inlet and outlet leg is enlarged, which means the curvature of the inner wall is strongly reduced and changes gradually compared to the baseline geometry.

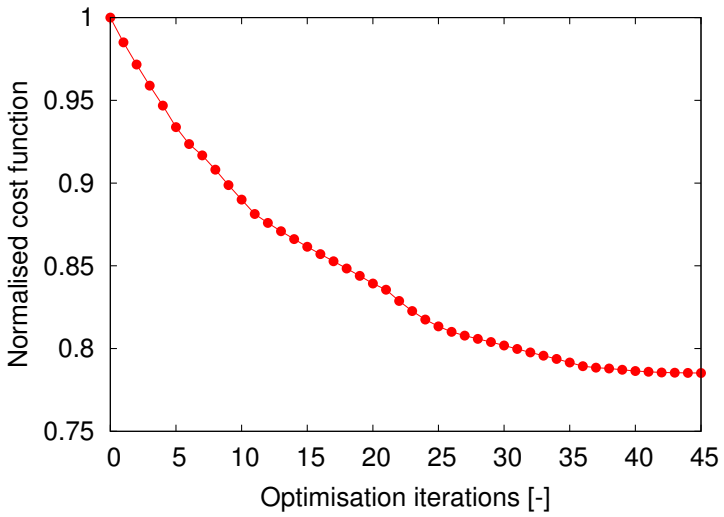
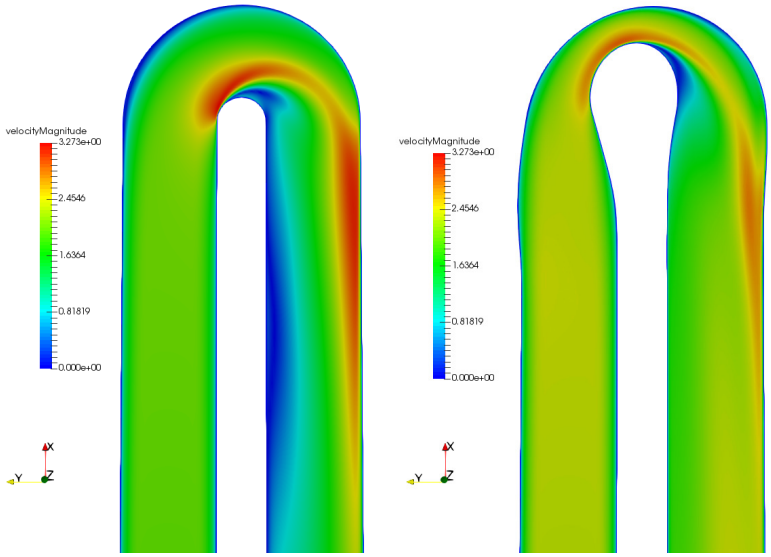


Figure 8.11: The cost function convergence history with the steepest descent method.



(a) Initial (b) Optimised

Figure 8.12: Velocity magnitude at middle plane before and after optimisation with the steepest descent method.

Above shape deformations reduce the abrupt change of velocity from the inner wall to the out wall, because when the most dominant force is related to the centrifugal force of particles following a bend streamline, and not the viscous forces, a good approximation is that the velocity gradient perpendicular to the streamline is proportional to the inverse of the curvature [46]. Therefore, the flow acceleration along the inner wall becomes milder such that the adverse pressure gradient along the flow direction is decreased, which has

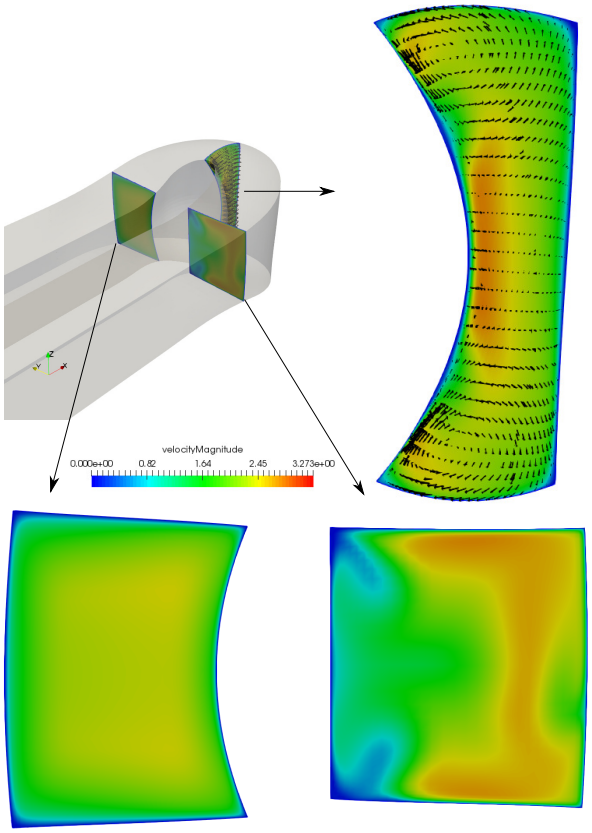


Figure 8.13: Velocity magnitude at different locations of the optimised geometry with the steepest descent method.

a positive effect on the reduction of flow separation. The result of this optimisation case is consistent with the experiments results on divider wall thickness reported by Liou et al. [261]. However, compared to the inner wall, the deformation of the outer wall is much smaller.

The velocity magnitude at three different locations are presented in Fig. 8.13. One can observe that the effective cross-section area in the outlet leg is increased due to the suppressed flow separation. The reduction of separation helps to reduce the total pressure loss. It can also be seen that, although the height is increased at the mid-bend, the width is decreased due to the convex shape of the inner wall. This convex shape splits the flow and drive it to the corner. As a result, the area with vortices is reduced such that they will produce less pressure losses. Also, lower flow velocity in the bend reduces the required pressure gradient to turn the flow.

Geometric results

During the optimisation, the continuity recovery steps presented in Section 5.5 are

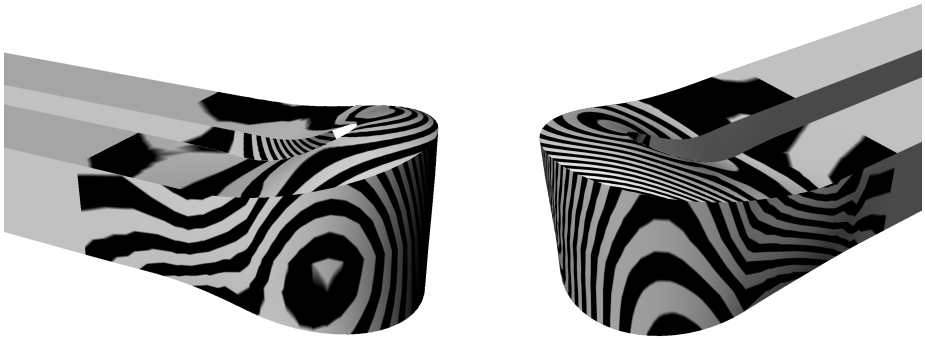


Figure 8.14: Zebra analysis for the optimised shape.

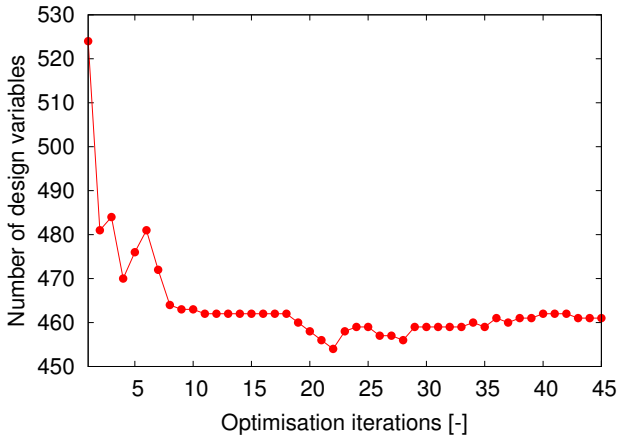


Figure 8.15: The number of design variables during the optimisation.

applied to recover the G_1 continuity. The recovery steps are controlled by a threshold. For this case, it is found that one recovery step can bring the G_1 deviation value below the chosen threshold value 10^{-5}. The results of Zebra analysis introduced in Section 7.6.3 for the optimised shape are presented in Fig. 8.14. As can be seen, the G_1 continuity is well maintained on edges where the G_1 continuity constraint is imposed.

The variation of the number of design variables, namely the size of $\boldsymbol{\alpha}$ is given in Fig. 8.15. This is because the constraint matrix is updated in each design iteration. Figure 8.16 presents the distribution of the singular values in first 10 iterations. It can be seen that the singular value distribution changes during the optimisation although the character remains, thus the effective ranks changes. The sum of the effective rank and the number of design variables is a constant as presented in Section 5.4.7. As a consequence, the number of design variables changes accordingly. Therefore, measures must be taken if one wants to utilise advanced optimisers, such as the BFGS approach, instead of the steepest descent method.

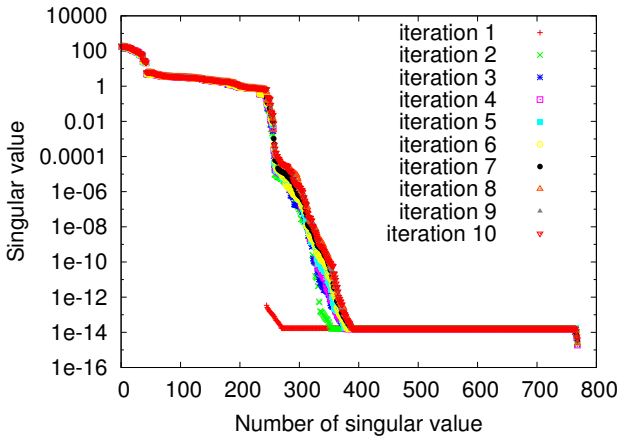


Figure 8.16: The distribution of singular values in first 10 iterations.

This optimisation is performed using a 64-bit, 8-core desktop, which has an Intel Xeon 3.40 GHz CPU and 16 GiB memory. The run time of each part in the optimisation is shown in Table 8.1. As can be seen, the flow and adjoint are most expensive parts, which cost around 91% of the runtime. The runtime of NSPCC, namely geometry reading, perturbation and writing, SVD and continuity recovery steps, together total sensitivity assembly and optimiser, only spend 7.49% of the total runtime. This indicates that, to perform CFD based shape optimisation more quickly, efficient flow and adjoint solver are essential. Note that in this case, only in the first iteration the flow and adjoint are fully converged as shown in Fig. 8.17, while from there on the one-shot method as described by Christakopoulos [61] is utilised to save computational cost.

Table 8.1: Computational cost breakdown for different parts of the optimisation loop with the steepest descent method.

Item	Run time (minutes)	Percentage
Adjoint	2842.62	61.61%
Primal	1355.45	29.38%
NSPCC and others	345.82	7.49%
Mesh deformation	38.71	0.84%
Flow sensitivity assembly	31.31	0.68%
Total	4613.91	100%
Average per iteration	102.53	

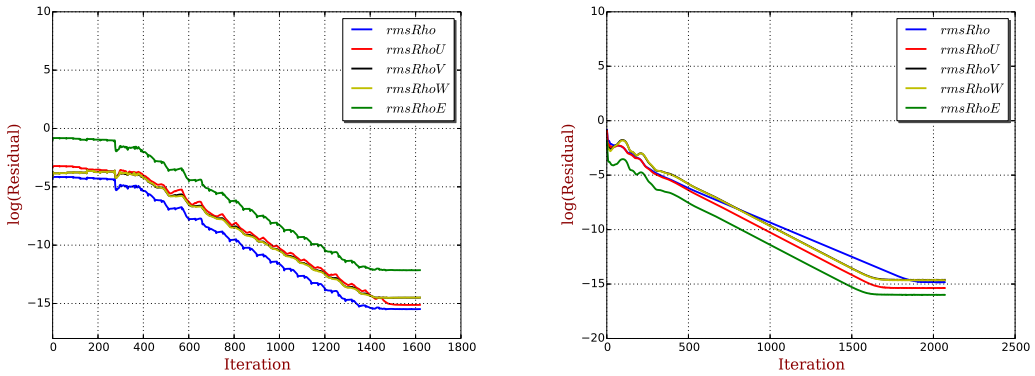


Figure 8.17: Convergence histories of flow (left) and adjoint (right) in the first iteration for U-bend case.

8.5.2 Optimisation results using BFGS

As proposed in Section 5.4.7, to apply the BFGS algorithm together with NSPCC, the design space need to be frozen for several steps. This idea is applied to the U-bend optimisation and results are reported in this section. To be precisely, the design space is frozen for every 4 iterations, i.e. the design space will be updated in the 1st, 5th and 10th iteration, and so on. Then BFGS is applied inside every 4 iterations with a fresh start. All other setting parameters are the same with the steepest descent case. Again, the weights of control points are frozen so the DoF is 640. Note that a third case with free weights will be presented in Section 8.5.3.

The comparison of convergence history of the cost function between BFGS and the steepest descent method is presented in Fig. 8.18. It can be observed that when BFGS is used as the optimisation in the way proposed in Section 5.4.7, the optimisation converges slight more rapidly and to a better result. Specifically, after 32 iterations the total pressure loss is reduced by around 23.1%, compared to only 21.5% in 45 iterations when the steepest descent method is employed.

Figure 8.19 presents the size of design variables during the optimisation, showing that the size of design variables is a constant during every 4 iterations. In addition, because the constraint matrix is fixed during these 4 iterations, the nullspace is also not changed. As a result, the deformation modes are frozen. Therefore, both the size of design variables and the deformation modes corresponding to them are the same, allowing the BFGS algorithm to be applied.

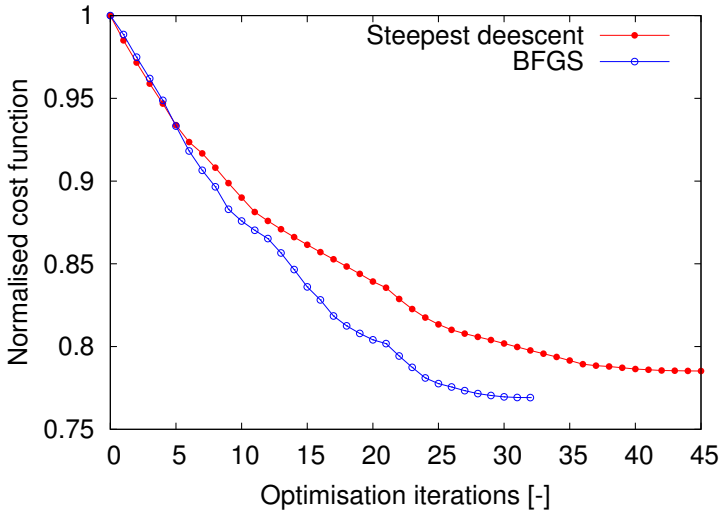


Figure 8.18: The cost function convergence history with the BFGS algorithm.

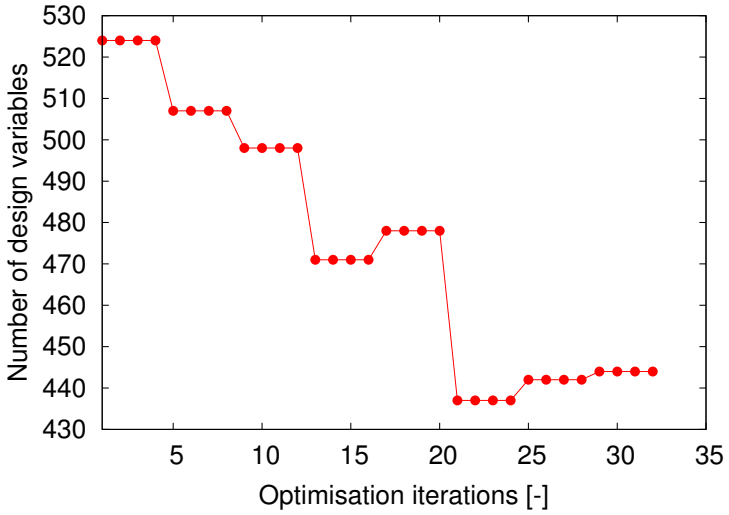
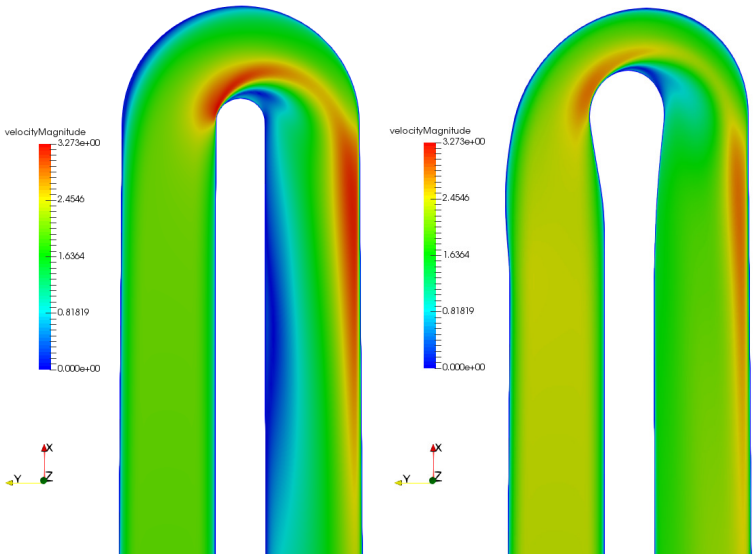


Figure 8.19: The number of design variables during the optimisation with the BFGS algorithm.

The velocity magnitude at the middle height plane is illustrated in Fig. 8.20, which indicates that the flow separation has been reduced significantly. Similar to the steepest decent case, both the thickness of the divider wall and the radius of curvature of the inner wall are increased after optimisation. Figure 8.21 presents the velocity magnitude at different location. As can be seen, the convex shape of the inner wall is also similar to the steepest descent case.



(a) Initial (b) Optimised

Figure 8.20: Velocity magnitude comparison before and after optimisation with the BFGS algorithm.

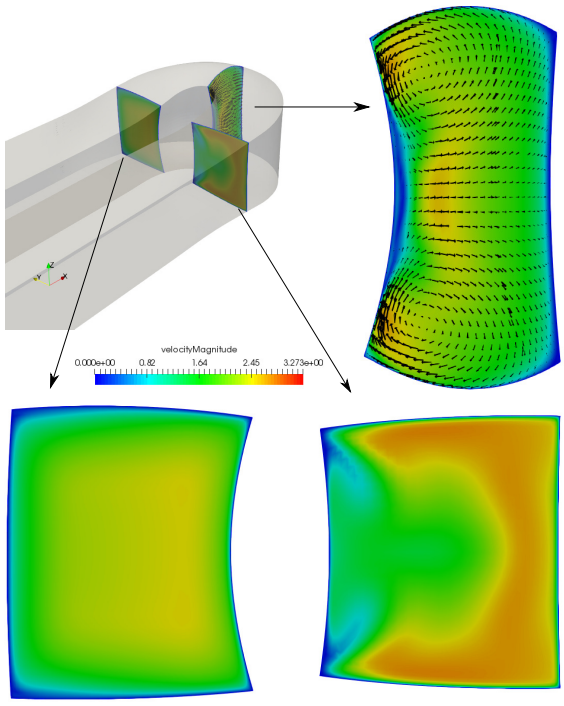


Figure 8.21: Velocity magnitude at different locations of the optimised geometry with the BFGS algorithm.

Physical expanation of the results

In this BFGS case, the deformation of the inner wall exhibits a similar pattern to that of the steepest descent case, but to a smaller extent. This can be seen clearly from Fig. 8.22, which shows the comparison of the geometry contour at the middle height plane. Therefore, the cross-sectional area in the 90 degree position of the U-bend is larger than the steepest descent case as shown in Fig. 8.23. As a consequence, the optimised bend with BFGS method has an enlargement in the first half which can reduce the flow velocity, and a contraction in the second half which accelerates the flow. This helps to additionally mitigate the adverse pressure gradient. In addition, the reduction of the velocity together with the increase of the radius of curvature leads to a reduction of the centrifugal force. This effect is crucial as it limits the tendency of the flow to move away from the inner wall. Therefore, the flow separation is reduced greatly.

Another aspect to be noted is that, the cross section shape shown in Fig. 8.13 leads to more friction loss because it is thinner and longer, compared to that illustrated in Fig. 8.21. Based on these discussions, one can see that the BFGS algorithm finds a better optimal shape than the steepest descent method.

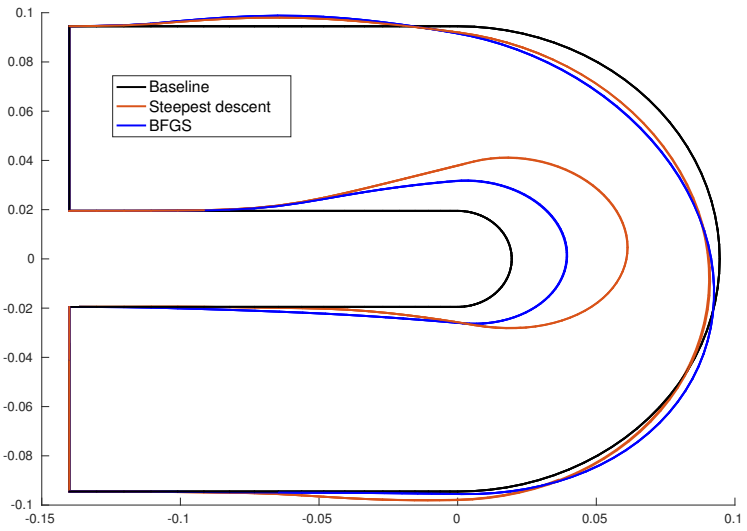
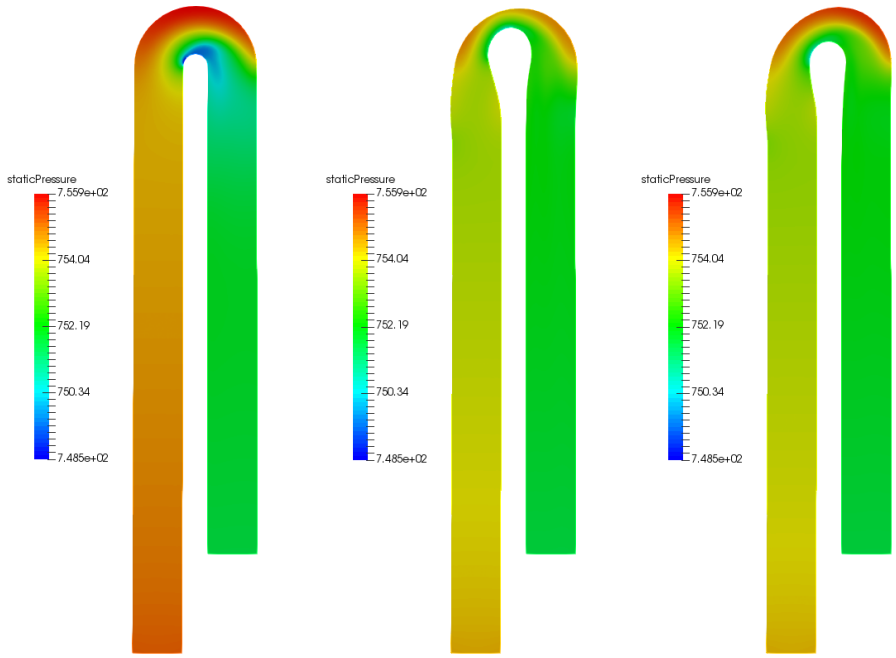
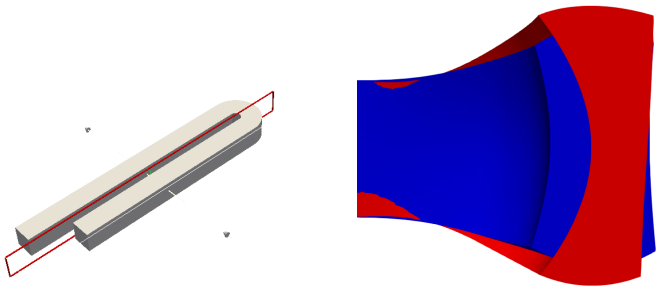


Figure 8.22: Comparison of geometry contour at middle height plane.



(a) Baseline (b) The steepest descent (c) BFGS

Figure 8.24: The comparison of pressure distribution before optimisation, and after optimisation with both optimisers at the middle height plane.



(a) The vertical cut plane (b) Optimised

Figure 8.23: The vertical cross section comparison between the steepest descent (red) and BFGS (blue) results.

The comparison of pressure distribution is shown in Fig. 8.24, which indicates that the pressure required at the inlet is reduced after optimisation. In addition, in both optimisation cases, the pressure difference between the inner and outer wall inside the turn is less than the datum bend. The smaller pressure gradient normal to the streamline direction implies weaker secondary vortices. Also, notice that the concentrated low

pressure region near the inner wall in the baseline U-bend corresponding to the high velocity area, is removed almost completely after optimisation.

The runtime of each part in this shape optimisation with BFGS method is presented in Table 8.2. Still, the flow and adjoint parts cost more than 94.5% of the runtime while all other components just contribute less than 5.5%. Compared to the steepest descent case, the total runtime of the optimisation is reduced. In addition, NSPCC costs less time because the design space is fixed in some iterations.

Table 8.2: Computational cost breakdown for different parts of the optimisation loop with the BFGS algorithm and fixed weights.

Item	Run time (minutes)	Percentage
Adjoint	2375.15	62.92%
Primal	1194.01	31.63%
NSPCC and others	153.69	4.07%
Mesh deformation	28.05	0.74%
Flow sensitivity assembly	24.19	0.64%
Total	3775.09	100%
Average per iteration	117.97	

8.5.3 Optimisation results with free weights and BFGS

In the first and second case presented above, the weights of control points are frozen during optimisation. In this section, weights are also added into the design space resulting in a total number of 864 DoF. The scaling strategy introduced in Section 5.6 is applied to homogeneous control points to make all components in similar order of magnitude. All the setting parameters are the same as in the frozen weights case presented in Section 8.5.2.

The cost function convergence history is shown in Fig. 8.25. It can be observed that the total pressure loss is reduced by around 25% after 35 iterations, larger than 23.1% reduction when the weights are fixed presented in Section 8.5.2, although 3 more iterations are performed.

Figure 8.26 presents the comparison of velocity magnitude at the middle plane before and after optimisation with the BFGS method. As can be seen, after optimisation the flow separation presented in the datum U-bend at this plane has been considerably reduced. Compared to the results with fixed weights, the separation here is much

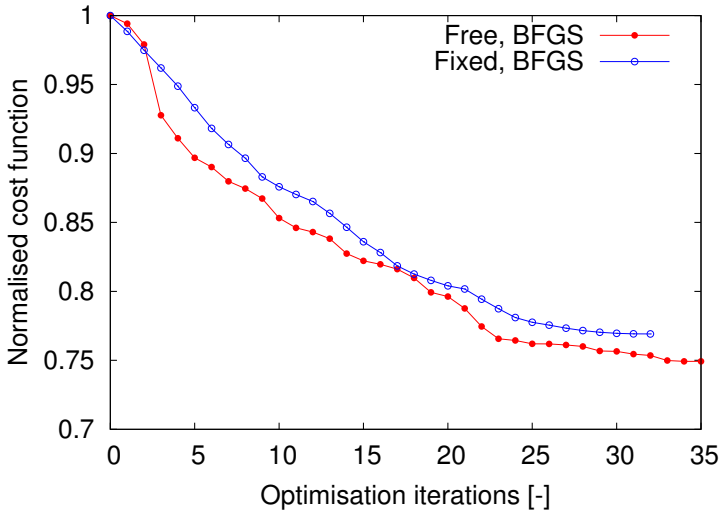
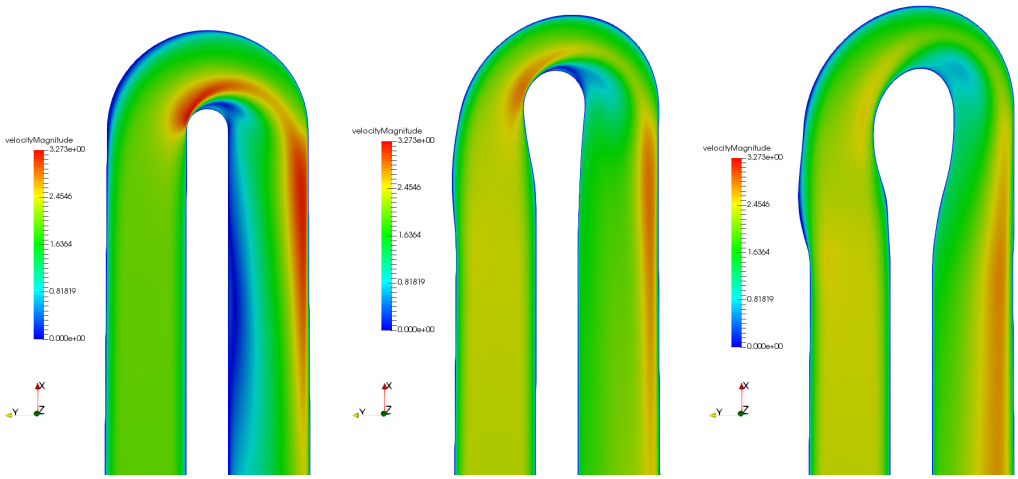


Figure 8.25: The convergence history of cost function with free weights.

smaller. Besides, the flow acceleration along the inner wall is milder now. The velocity magnitude at five different locations are shown in Fig. 8.27, from which one can see that the effective area in the second half of the bend is increased significantly due to the reduced separation. However, at the top and bottom of the inlet leg, some separations appear which are not observed in Section 8.5.1 and Section 8.5.2.



(a) Initial (b) BFGS with fixed weights (c) BFGS with free weights

Figure 8.26: Velocity magnitude comparison before and after optimisation with the BFGS method.

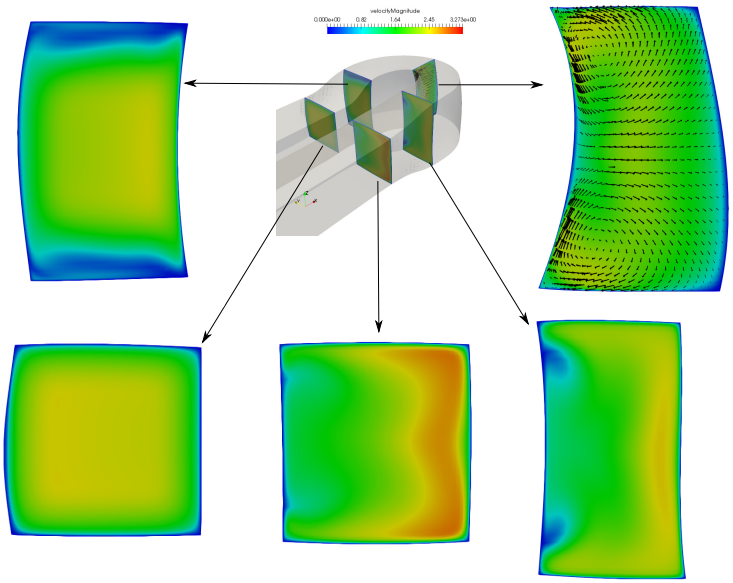


Figure 8.27: Velocity magnitude at different locations of the optimised geometry with free weights.

Physical expanation of the results

Different flow behaviours and pressure drop is due to some noticeable geometry differences which can be observed compared to the case of frozen weights. Firstly, the radius of curvature of the inner circle is strongly increased, in both the forward and return channel. Compared to the fixed weight cases, the deformation of the inner wall of the outlet leg here leads to a stronger contraction after the turn, which helps to reduce the adverse pressure gradient to a larger extent. The convex inner wall in the second half of the bend also fills the space which is occupied by the separated flow in the baseline geometry. Secondly, as can be seen from Fig. 8.27, the forward channel is enlarged significantly which helps to lower the velocity in the inlet leg. The return channel on the other hand is strongly contracted, thus the channel exhibits a more profound diverging-converging shape, which helps to reduce the separation as explained in Section 8.5.2.

Another feature of the optimised U-bend is that, the outer wall also has considerable deformation. To be precisely, the outer wall is pushed outwards almost everywhere compared to the datum geometry, which can be observed clearly from Fig. 8.28. In combination with the increase of radius of curvature of the inner wall, the enlarged outer wall leads to smaller difference between the velocity on the internal and external wall. Therefore, the flow acceleration in the first half of the bend along the inner wall and the deceleration in the second half are smaller compared to the case when weights are fixed,

reducing the flow separation in the outlet leg further.

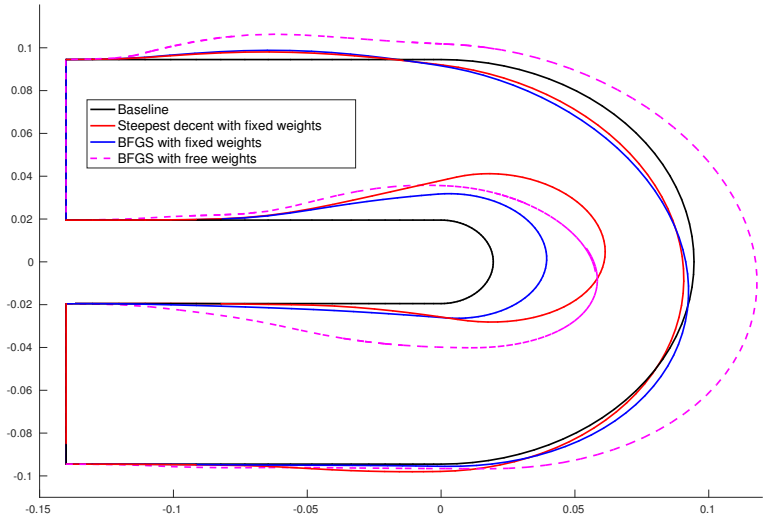
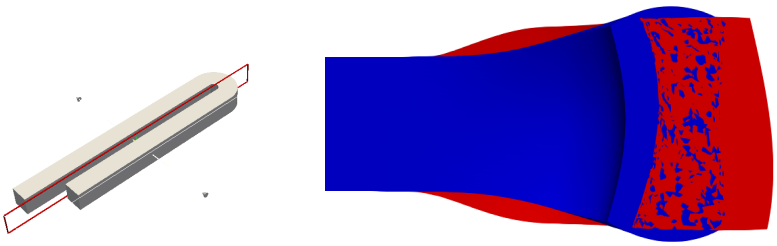


Figure 8.28: Comparison of geometry contour at middle height plane with free weights.

Figure 8.29 illustrates the optimised cross section shapes at 90 degree position obtained the BFGS algorithm. It can be seen that when the weights are added into the design space, the cross sectional area is larger than in the frozen weights case. Also, when weights are free the width of the cross section is larger while the height is slightly smaller, which helps to reduce the friction loss.



(a) The vertical cut plane (b) Optimised

Figure 8.29: The vertical cross section comparison between the fixed (blue) and free (red) weights case.

The runtime of each part of this case is presented in Table 8.3. As can be seen, the flow and adjoint parts cost more than 94% of the runtime while all other components just occupy less than 6%.

Table 8.3: Computational cost breakdown for different parts of the optimisation loop with the BFGS algorithm and free weights.

Item	Run time (minutes)	Percentage
Adjoint	2307.94	59.51%
Primal	1340.70	34.57%
NSPCC and others	173.06	4.46%
Mesh deformation	32.15	0.83%
Flow sensitivity assembly	25.70	0.63%
Total	3878.24	100%
Average per iteration	110.81	

Comparison with experimental data

Coletti et al. [27] also conducted experiment using their optimised U-bend geometry. It would be helpful to compare the simulation results and experimental data. Figure 8.30 presents the comparison of results obtained in this work with those from the experiment, as well as the initial flow. It can be seen that in both cases, the flow separation has been successfully reduced, although not completely removed. In addition, the results obtained in the present study has smaller separation after optimisation.

Regarding the geometry, the inner walls show similar defomation behaviour in both cases. Firstly, the space in between the forward and return channel increased. Secondly, the radius of curvature of the inner circle became larger. These behaviours are regarded as main reasons for separation reduction in the U-bend case [46, 256, 261], which in turn is the reason for reduction of pressure loss.

However, the deformations of the outer wall were different, although the outer walls were pushed outwards in both cases. The main reason is that, 2D shape optimisation was performed for the U-bend in [26], thus the width of bend part firstly increase and then decrease in order to reduce the adverse pressure gradient, as shown in the right of Fig. 8.30. In the present work, 3D shape optimisation was performed, it is the cross section area that firstly increase and then decrease instead of the width. Another possible reason is that different parametrisations were utilised. In the present study, NURBS was used to describe the geometry and the number of DoF was 864. The U-bend in [27] was parametrised with piece-wise Bezier curves and there were 26 DoF. Other reasons include different solvers and different optimisers.

Above comparison between numerical and experimental results indicates that the

optimised shape exhibits similar features as those shown in experiments, thus demonstrates to some extent that the results in this work are reliable.

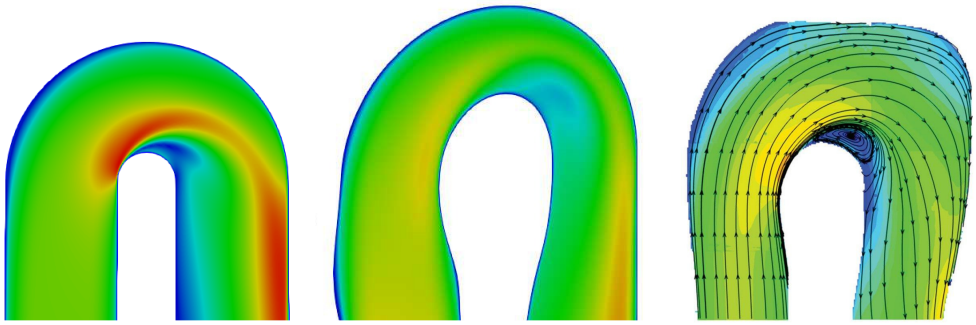


Figure 8.30: Comparison of velocity magnitude at the middle height plane of initial geometry (left), optimised geometry between STAMPS results (middle) as well as experimental results (right).

8.6 Summary

In this chapter, automatic numerical shape optimisations have been performed for a U-bend cooling channel, aiming at reducing the pressure drop. The feasibility of the CAD-based shape optimisation framework consists of the NSPCC kernel and a compressible flow solver as well as a discrete adjoint solver, has been demonstrated.

The U-bend geometry is parametrised with NURBS. Different continuity levels are imposed to different edges to satisfy the geometric continuity constraints. It has been shown that the geometric continuity can be effectively maintained in the optimal geometry. The effective rank proposed in this work, namely determining the rank based on $\mathbf{C}\mathbf{C}^T$ instead of $\mathbf{C}$, is utilised to determine the design space, which has been shown again as a reasonable pre-conditioner.

As a first case, the steepest descent method is utilised since the number of design variables is not a constant if the constraint matrix is updated in each design step. In a second case, the design space is fixed for several iterations, making it possible to couple NSPCC with the BFGS algorithm. Although both optimisers successfully find new U-bend geometries which have lower pressure loss, it has been shown that the BFGS method can give a better shape in fewer iterations. This is due to that BFGS is a quasi-Newton method, which is more powerful than the steepest descent approach.

In a third case, apart from position of control points, the weights attached to control points are also added into the design space during the optimisation. Therefore, a larger

design space is obtained. As a consequence, the pressure drop reduction is larger than that of fixed weights case, and the flow separation is better suppressed in the optimised bend.

It should be mentioned that one of the objectives of this test case is to demonstrate the advantage of NURBS over B-splines in two aspects. Firstly, NURBS can represent circular shape precisely with few control points. Secondly, NURBS provide weights thus offering more DoF. Therefore, only 4 control points are used to represent the circular part of the bend. However, one can refine the control net to use more control points, which could possibly give better shape.

Chapter 9

Aerodynamic design optimisation of the ONERA M6 wing

9.1 Introduction

The ONERA M6 wing [179] is a well-known testcase for transonic flow studies in computational aerodynamics, and it has been used by numerous authors to demonstrate various methods [50, 121, 262–265]. For example, Martin et al. [50] optimised the M6 wing at inviscid flow conditions to demonstrate their CAD-based shape optimisation framework. The M6 wing was described with B-spline surfaces. However, the paper did not discuss whether the weights were perturbed or not during optimisation. Liang et al. [121] ran a multi-objective optimisation for the M6 wing with gradient-free methods. Maters et al. [264] demonstrated the three-dimensional subdivision parametrisation with the M6 wing.

In this work, The NSPCC CAD kernel is coupled with the in-house flow and adjoint solver STAMPS and a gradient-based optimiser to minimise the drag of the ONERA M6 wing in transonic Euler flow conditions. The aim is to demonstrate the effectiveness of the NSPCC-based optimisation framework in transonic aerodynamic shape optimisation. In addition, the effect of parametrisation on final results are also investigated. To this end, the ONERA M6 wing is re-parametrised with NURBS surfaces including weight adjustments to represent the three dimensional wing accurately, resulting in fewer control points and smoother variation of curvature. Optimisation are performed for both the B-Spline and NURBS parametrisations. Besides, a method to impose thickness constraint for the wing is developed and demonstrated. The parametrisation of the wing is

introduced in Section 9.2. The drag minimisation results are presented and analysed in Section 9.3.

It should be mentioned that the aim of this study is not to find the best geometric representation of the geometry or the minimum set of design variables, but to demonstrate the feasibility of the optimisation framework based on NSPCC in transonic flow.

9.2 Parametrisation

9.2.1 B-splines representation of ONERA M6 wing

The M6 wing is a typical example of a geometry that cannot be represented exactly in CAD, hence defining a CAD model for the M6 wing requires a trade-off between fidelity and complexity. The M6 wing approximated with B-splines used in this chapter was created by a former member [180, 250] of the CFD group at QMUL, according to the description by Schmitt et al. [179]. The shape and control points of the B-spline M6 wing are shown in Fig. 9.1. This M6 wing consists of two surfaces, namely the upper and lower surface. For each surface, there are 12 control points in the spanwise direction and 13 control points along the chordwise direction. So in total there are 312 control points.

The shape after perturbing one control point in vertical direction is given on the right of Fig. 9.1. As can be seen, only a portion of the surface is modified, leaving the rest intact. This clearly shows the local modification property of B-splines.

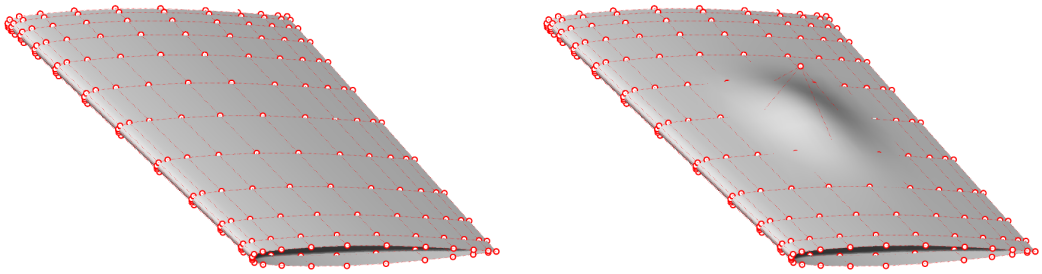


Figure 9.1: B-spline ONERA M6 wing and its control points: original (left) and perturbed (right).

9.2.2 M6 wing approximation using NURBS

In this section, the B-spline M6 wing displayed in Fig. 9.1 is re-parametrised with NURBS. The obtained geometry will then be optimised to improve its aerodynamic performance.

Lepine et al. [109] fitted NURBS for 2D aerofoils and showed that the number of control points can be reduced with NURBS. Based on this fact, in this work the 3D wing shape is directly approximated with NURBS within the NSPCC framework. The approximation has two major advantages. Firstly, even though the cost of computing the derivatives is constant with the adjoint approach, the lower number of control points will make it easier for optimiser to converge the KKT system. Secondly, and more importantly, the smaller number of control points will result in a smoother variation of curvature along the profile, which is an essential quality in the design of transonic wings and turbo-machinery blades.

As a first step, the datum M6 wing shape shown in Fig. 9.1 is approximated using NURBS representations. A set of wings with different numbers of NURBS control points produced from the Rhinoceros[1] are used as starting points. The cost function of this optimisation problem is defined as the root-mean-squared difference between initial and target geometries, i.e.:

$$J = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{Initial_i} - \mathbf{X}_{Target_i})^2}, \tag{9.1}$$

where $\mathbf{X}_{Initial_i}$ and $\mathbf{X}_{Target_i}$ are surface points sampled on the initial and target wing geometry, respectively. N is the number of sample points. To improve the surface fidelity in regions of high curvature, a cosine spacing as shown in Fig. 9.2 is used which concentrates the sampling points near the leading edge than near the trailing edge, as shown in Fig. 9.2.

The cosine spacing is implemented as following:

1. Evenly choose n points between 0 and $\frac{\pi}{2}$, i.e. $\beta_i = \frac{\pi}{2} \cdot \frac{i}{n-1}, i = 0, 1, ..., n-1$
2. Along the chordwise direction in parametric space, choose n points u_i satisfying $u_i = 1.0 - cos(\beta_i)$. As a consequence, u_i will be in the range of $[0, 1]$, and there will be more points near the leading edge, and less points near the trailing edge, as indicated in Fig. 9.2.

[1] https://www.rhino3d.com/

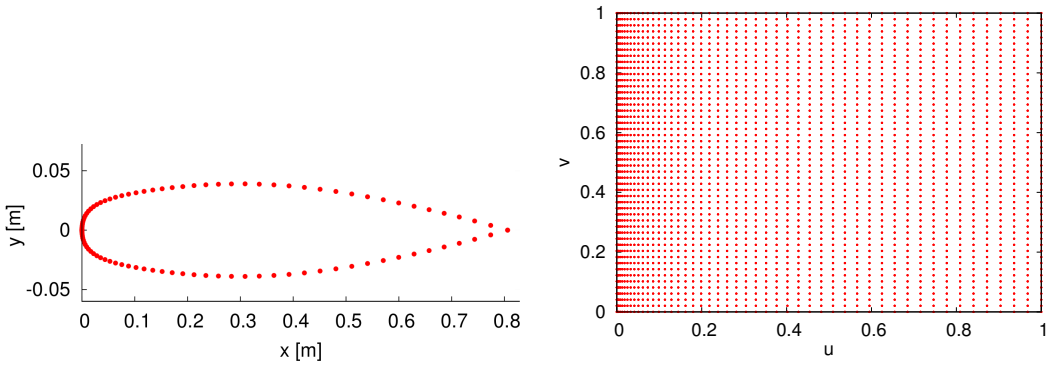


Figure 9.2: Cosine spacing sample points. Left: points of one section. Right: all points in parametric space

For the M6 fitting in this study, 5000 surface points are sampled on each geometry, 100 points in the chordwise and 50 points along the spanwise direction.

The initial wing geometries with different number of NURBS control points are driven towards the target shape by performing optimisation using the L-BFGS method [133, 134]. The derivative of the objective function w.r.t. the design variables, i.e. the 2-D coordinates of the control points in the profile plane and their weights, are calculated using AD. Note that during the optimisation, the control points at the leading and trailing edge are fixed for the purpose of fixing the chord length and angle of attack (AoA). The second column of control points at both sides near the leading edge can only move vertically (y direction) to maintain the continuity. Other control points can move both vertically and along the chordwise direction (x direction). In addition, weights of control points are also changeable.

The objective function values in equation (9.1) resulting from the optimisation with varying numbers of control points is given in Fig. 9.3. It can be observed that the objective value continues to drop with increased number of control points. We use the typically accepted value of 10^{-4} to determine an acceptable fit.

The M6 wing described using 26 B-spline control points and 16 NURBS control points are illustrated in Fig. 9.4. The comparison of the root section is shown in Fig. 9.5. It can be seen that the two wing sections match well. The results demonstrated that by using NURBS, a M6 wing with fewer control points is obtained.

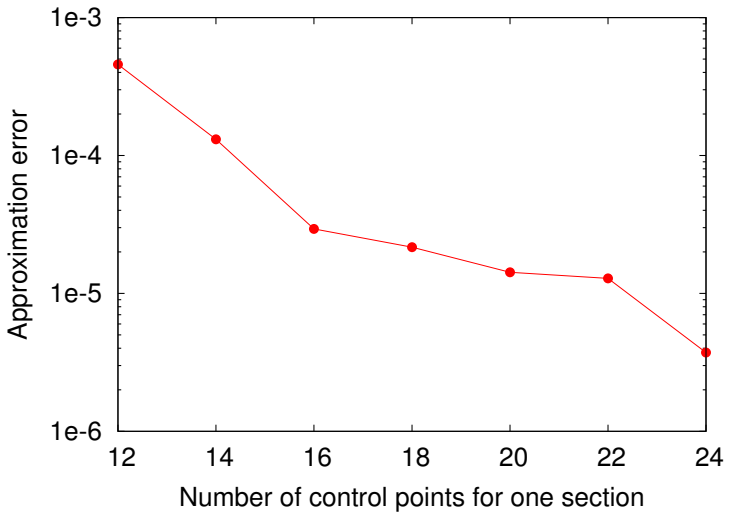


Figure 9.3: Fidelity of NURBS approximation with varying number of control points.

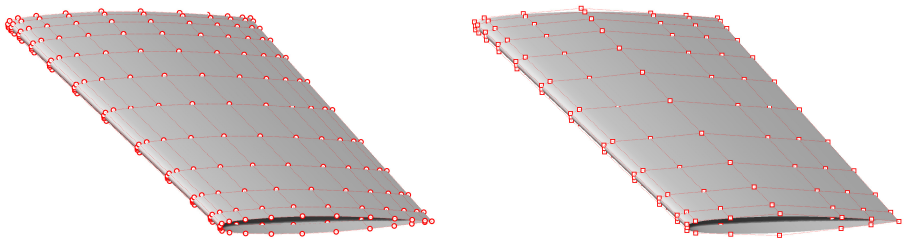


Figure 9.4: ONERA M6 wing with 26 B-spline (left) and 16 NURBS control points (right).

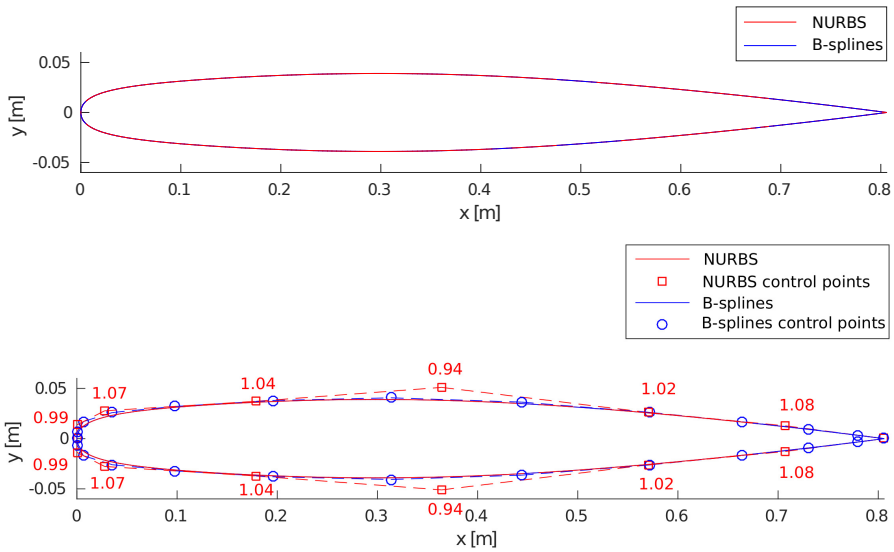


Figure 9.5: Comparison of ONERA M6 wing root section using 26 B-spline and 16 NURBS control points. Upper: shape of profile. Lower: shape of profile and control points and optimised weights.

9.3 Drag minimisation of the M6 wing

To investigate the effect of parametrisation on optimisation results, both the B-spline and NURBS M6 wing described above are utilised as starting point of the drag minimisation in the transonic regime and inviscid flow condition. The steepest descent method with Armijo line search introduced in Section 3.3 is employed as optimiser.

9.3.1 Mesh and case set up

The tetrahedral mesh of ONERA M6 wing used in this work, which contains 135,204 nodes and 771,129 tetrahedral elements, is shown in Fig. 9.6. It should be noted that in the current mesh the shock area is not refined, since this case serves as a proof-of-concept test for the CAD-based optimisation chain and its functionalities. This can be improved in future work to capture the shock wave better.

Figure 9.7 shows the the upper surface in parametric space after mesh projection as presented in Section 5.3. Throughout the optimisation process, the parametric coordinates (u, v) are considered constant, ensuring the mesh points always lie on the CAD geometry surface.

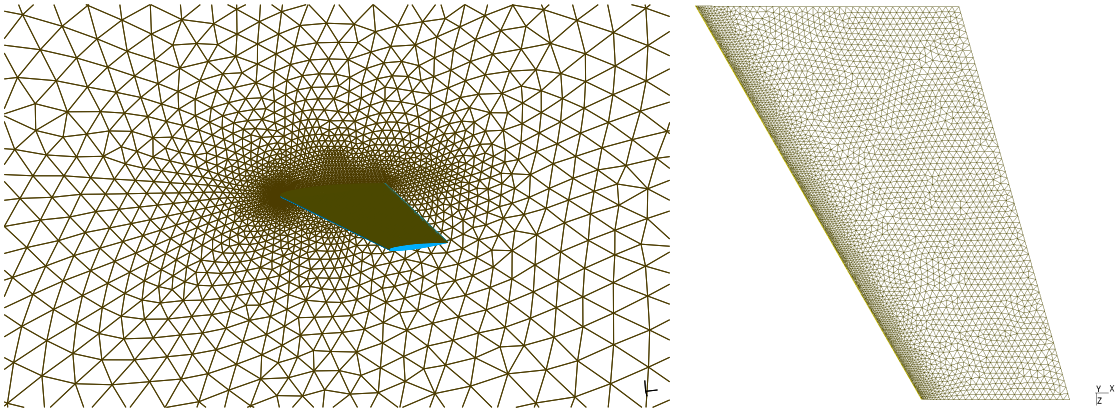


Figure 9.6: Mesh of ONERA M6 wing

The in-house compressible solver STAMPS introduced in Section 4.2.2 and 4.3.3 is used to perform flow analysis and provide the sensitivity of the objective function w.r.t. surface mesh node positions, $\frac{\partial J}{\partial \mathbf{X}_s}$. Case informations are listed as following:

- Flow conditions:

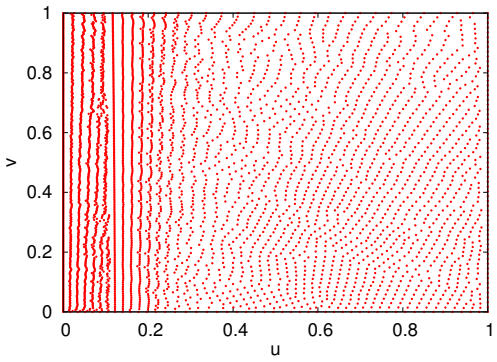


Figure 9.7: The upper surface of M6 wing in parametric space.

  - Inviscid flow
  - Freestream Mach number $Ma = 0.84$
  - Freestream temperature $T_\infty = 300K$
  - Angle of attack (AoA) $= 3.06°$
  - Pressure $p_\infty = 101325Pa$

- Mesh deformation method: inverse distance weighting (IDW) [177].
- Solver: JT-KIRK solver, Roe flux, 2nd order.
- Computing platform: a 64-bit and 8-core desktop, with an Intel Xeon 3.40 GHz CPU. The memory is 16 GiB.
- Control points at the leading and trailing edge are frozen to fix the AoA. The second column of control points at both sides near the leading edge can only vertically to maintain the continuity. Other control points can move vertically and along the chordwise direction. Weights of control points can also be added into the design space, if required. It should be mentioned that control points are used as design variables in this case, because the test point approach to impose geometric continuity it not utilised.
- Objective function: drag with lift constraint, defined as:

$$J = D + c * (L - L^*)^2, \tag{9.2}$$

  where D is drag force, L is lift force, and L^* is initial lift, which is 11104.21 N here. c is the penalty coefficient chosen as 0.0001, but this value can be adjusted

to change the importance of the lift force change on the objective function.

Besides, the one-shot methodology [175, 176] is utilised to accelerate the convergence of the solver.

9.3.2 Scaling method

The scaling method is needed if both position and weights are added into the design space. In this chapter, the idea of scaling proposed in [110, 266] is used. The aim is to create dimensionless design variables in the range $[-1, 1]$.

The scaled variables are:

$$X_i = \frac{x_i - x_i^0}{a \cdot chord} \tag{9.3}$$

$$Y_i = \frac{y_i - y_i^0}{a \cdot t_{max}} \tag{9.4}$$

$$Z_i = \frac{z_i - z_i^0}{a \cdot span} \tag{9.5}$$

$$\Omega_i = \frac{1}{ln(b)} ln\left(\frac{\omega_i}{\omega_i^0}\right) \tag{9.6}$$

where a and b are positive constant, $chord$ is the chord length of the root section, t_{max} is the maximum thickness of the root section, $span$ is the span of M6 wing. $x_i^0, y_i^0, z_i^0, \omega_i^0$ are initial coordinates and weights, respectively. The normal value of a is 0.05, such that the variation of the scaled variables between ± 1 yields $\pm 5\%$ variation of the chord length for the x variables, $\pm 5\%$ variation of the maximum thickness for the y variables and $\pm 5\%$ variation of the span for the z variables, respectively. Regarding the weights, their effect on the wing is better captured by a logarithmic representation than a linear one. The typical value of b is 2, such that the variation of Ω_i between ± 1 leads to the variation of weights between $0.5\omega_i^0$ and $2\omega_i^0$. This logarithmic scaling also help to avoid negative weights. Other values can also be used during the optimisation process, for instance use $a = 0.1$ and $b = 4$.

Then scaled control points $(X_i, Y_i, Z_i, \Omega_i)$ are then utilised during the optimisation. Note that in this study, the control points are frozen in z direction. The derivatives of the surface points w.r.t. these scaled control points are computed using the AD tool Tapenade in the NSPCC kernel.

The CAD sensitivity contour of the displacement in y direction of six control points on the upper surface are presented in Fig. 9.8. One can see clearly that one control point

only affects a small area of the surface. The CAD sensitivity verification for ONERA M6 wing is presented in Section 6.2.3.

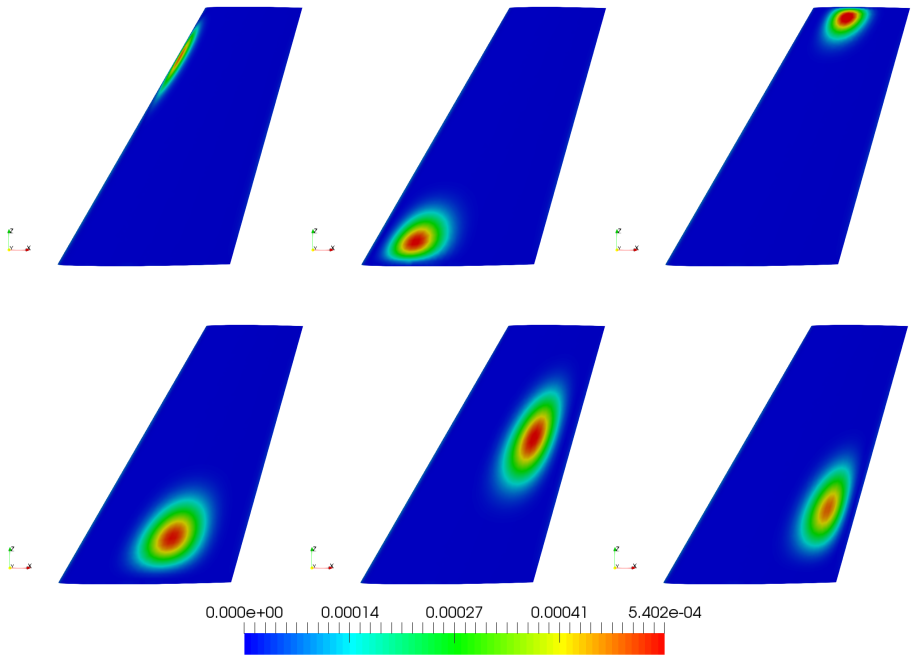


Figure 9.8: CAD sensitivity contours for ONERA M6 wing.

9.3.3 Initial flow field

The pressure contours and Mach number of the baseline geometry are shown in Fig. 9.9 and Fig. 9.10, respectively. As can be seen, the typical 'lambda' shock on the top surface is very clear.

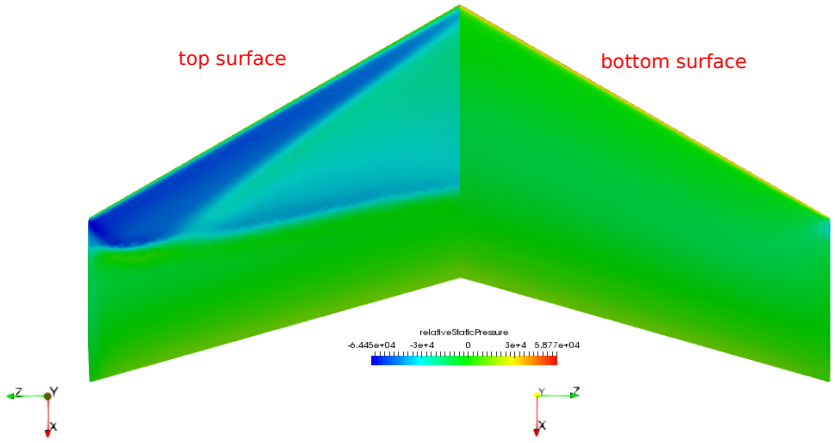


Figure 9.9: Baseline geometry pressure contours on top and bottom surface.

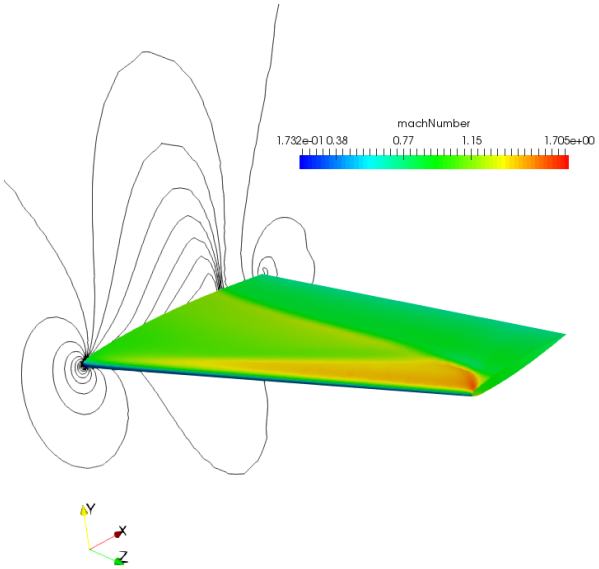


Figure 9.10: Mach number distribution of the baseline geometry and isolines at the symmetry plane.

Solver validation and mesh convergence study

As mentioned in Section 4.2.2, developers of STAMPS have performed validation for ONERA M6 wing and demonstrated that STAMPS was capable of providing valid solutions for this case. Actually, Christakopoulos [61] has also conducted a mesh convergence study, by using four different mesh levels. In that mesh convergence study, flow conditions were the same as in the present study. As can be seen, for different levels of mesh, the results show good match with experimental data. More details on the mesh convergence study could be found in [61].

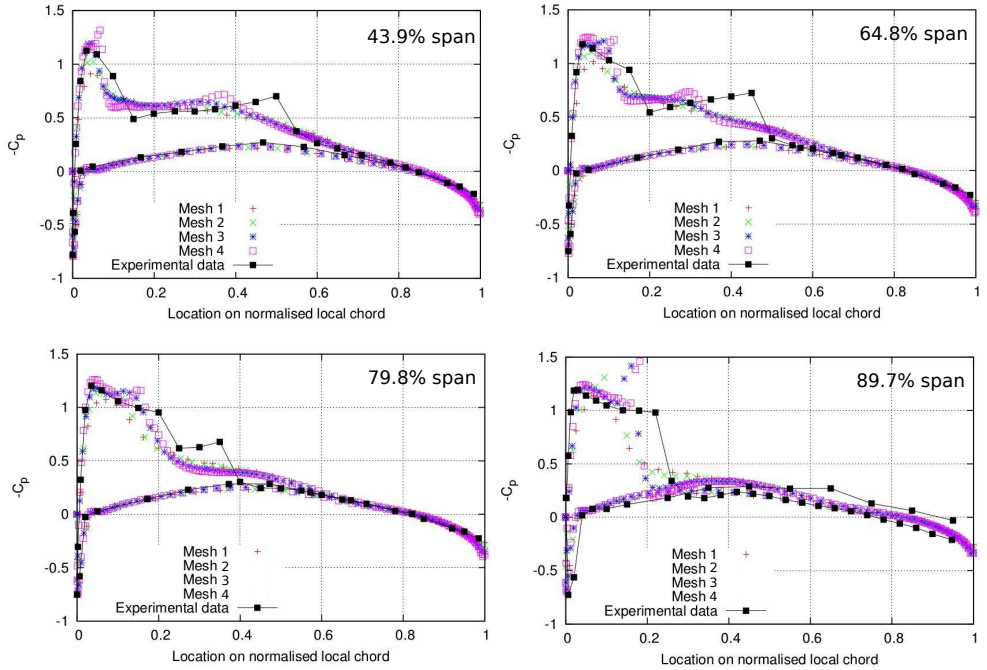


Figure 9.11: Pressure coefficient distributions in ONERA M6 wing mesh convergence study [61].

Comparison with published solutions

The ONERA M6 wing is a well-known and widely used test case, thus it would be very helpful to comparison solutions given by STAMPS and those in the literature. Agarwal et al. [96] performed CAD-based shape optimisation for the ONERA M6 wing, by utilising the open source solver SU2 to solve the compressible Euler flow equations. In that study, an unstructured mesh with 154,617 nodes and 707,115 tetrahedral elements was utilised, which is very similar to the mesh density used in the present work. Mach number and angle of attack are also the same. Fig. 9.12 illustrates the comparison of the top surface pressure contours of the baseline geometry between the present study and Agarwal's work. As can be seen, in both results the shock on the upper surface are very clear, and both results show similar behaviour. This is also similar to results available from more studies in the literature [121, 262, 265].

Based on the comparison with experimental results as well as results from the literature, one can see that STAMPS is able to provide reliable results for the ONERA M6 case.

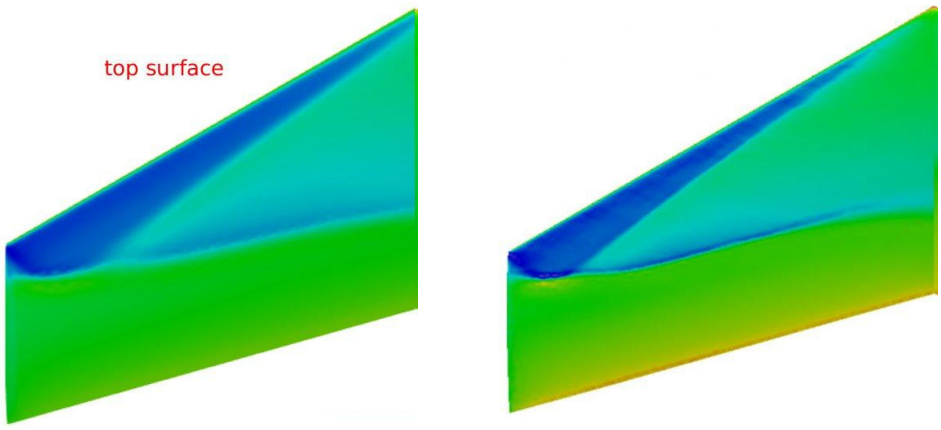


Figure 9.12: Comparison of baseline geometry pressure contours between the present study (left) and the literature (right) [96].

9.3.4 Optimisation results

9.3.4.1 Effect of scaling

The scaling method presented in Section 9.3.2 is applied to the optimisation of the NURBS M6 wing. To investigate the influence of the scaling on optimisation results, an optimisation without scaling is also performed. The comparison of the cost function convergence histories is shown in Fig. 9.13. As can be seen, if the scaling method is not used, the cost function exhibits a smaller reduction with a slower convergence rate. The comparison demonstrates the effectiveness of the scaling method utilised in this study. Therefore, the scaling strategy is employed in optimisation with free weights.

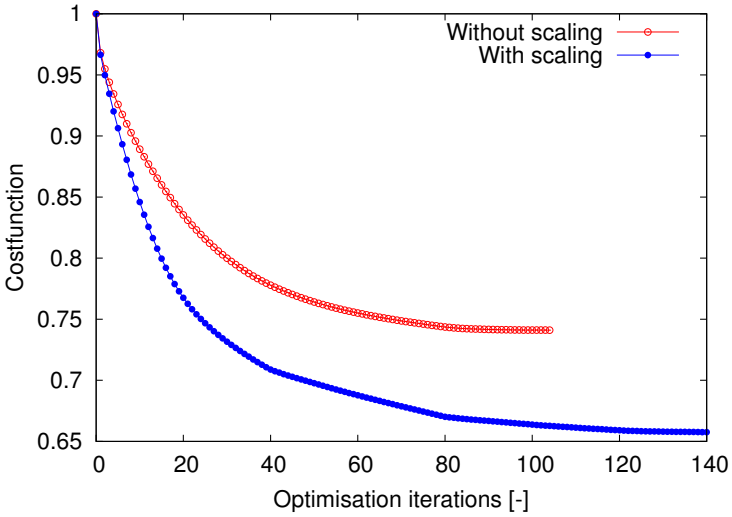


Figure 9.13: Cost function convergence histories with and without scaling.

9.3.4.2 Comparison of results between B-splines and NURBS

The optimisation results for both the B-spline and NURBS M6 wing will be presented and compared in this section.

The comparison of objective function during the optimisation are shown in Fig. 9.14. As can be seen, the cost function reduction of the NURBS case is larger than that of the B-spline case, although more iterations are needed. The values of drag and lift coefficients as well as the objective function values are listed in Table 9.1. In the NURBS case, the drag is reduced by around 36.0%, which is larger than that of 34.2% in the B-spline case. However, in the NURBS case, the optimised wing loses 2.66% of lift, which is larger than the B-spline case (2.1%). The loss of some lift is as expected, because the penalty part in the objective function is not very strict such that more drag reduction can be achieved.

Table 9.1: Drag and lift coefficients, and objective function value.

Case	Drag coefficient	Lift coefficient	Objective function (scaled)
Initial	0.01281	0.28052	0.01281
B-splines	0.00843	0.27464	0.00859
NURBS	0.00828	0.27307	0.00843

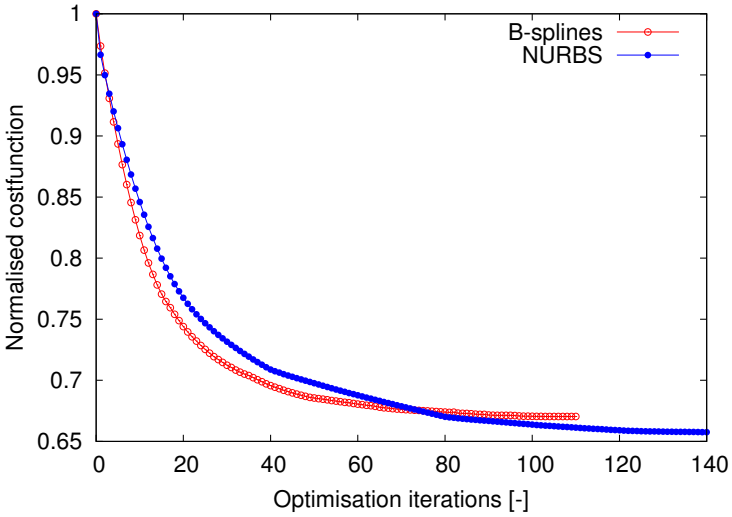


Figure 9.14: Cost function convergence histories with B-splines and NURBS.

The comparison of pressure contours on upper surface after optimisation is presented in Fig. 9.15, which clearly indicates that in both cases the shocks on the upper surface are reduced significantly. This is also well illustrated on the left of Fig. 9.16 where the pressure coefficients (C_p) on both upper and lower surface at different spanwise positions are given.

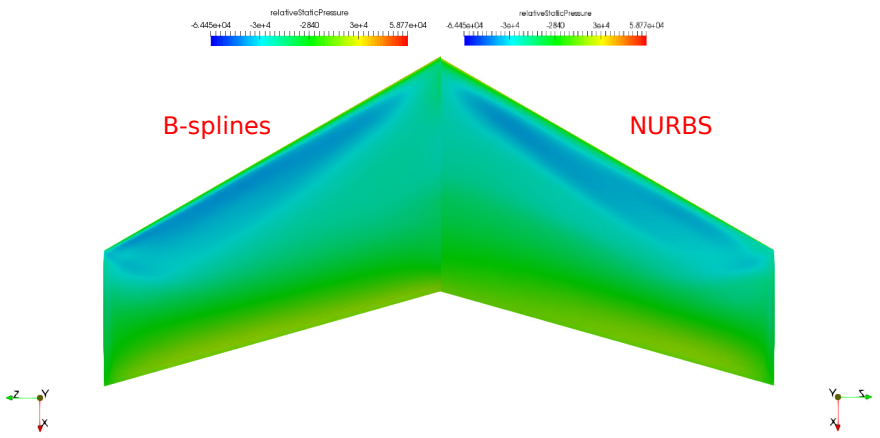
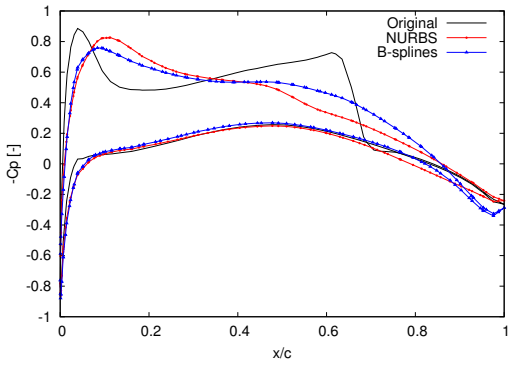
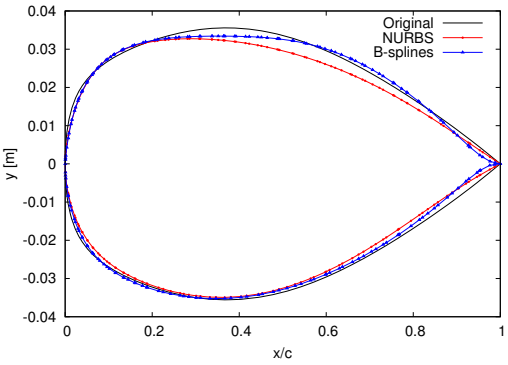


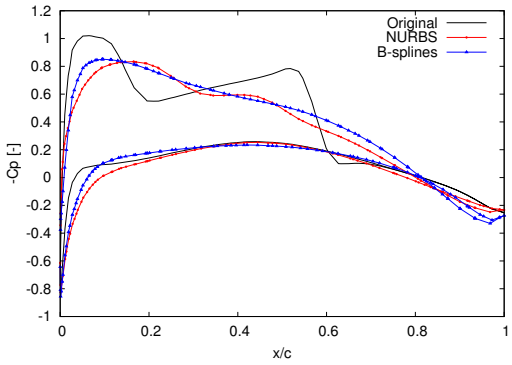
Figure 9.15: Comparison of pressure contour on upper surface after optimisation with different number of control points.



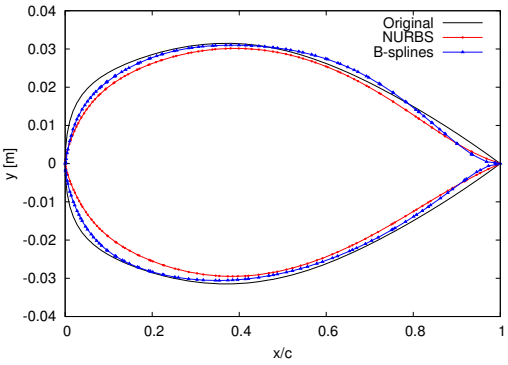
(a) 19.9% span: C_p



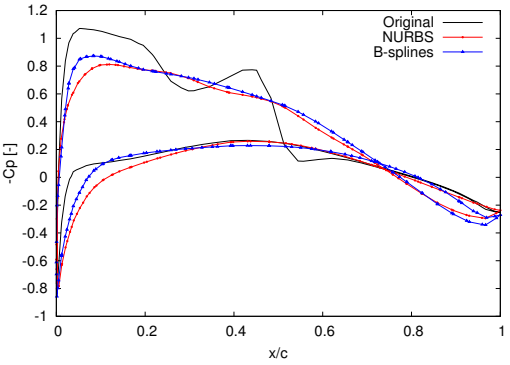
(b) 19.9% span: section shape



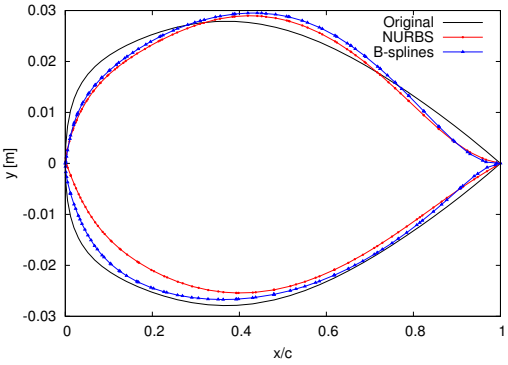
(c) 43.9% span: C_p



(d) 43.9% span: section shape



(e) 64.8% span: C_p



(f) 64.8% span: section shape

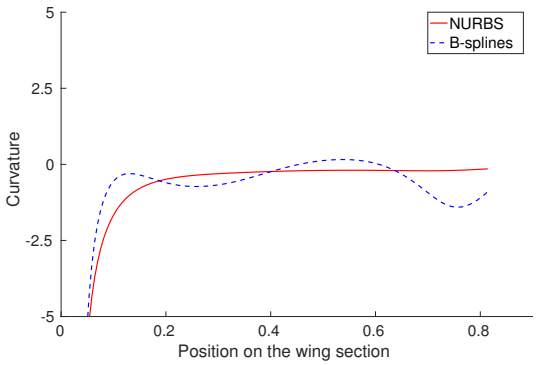
Figure 9.16: Comparison of cross section and pressure coefficient at different span-wise positions.

The comparison of wing profiles at different spanwise positions are given on the right of Fig. 9.16. Although the optimised shapes in both cases become thinner near the trailing edge, the B-spline case has smaller thickness and exhibits a large curvature variation there. A clearer comparison of profile curvature is presented in Fig. 9.17, which shows

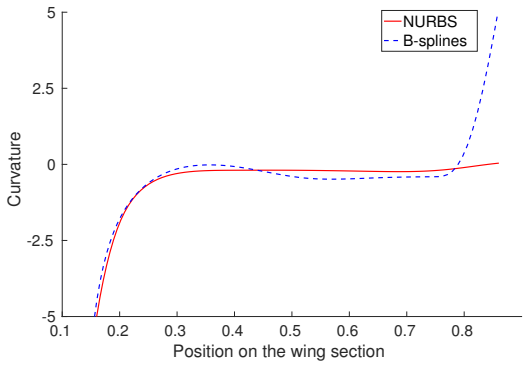
the comparison of curvature of four wing sections at different spanwise locations. The curvature of a parametric curve at one point is defined as [267]:

$$\kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}} \tag{9.7}$$

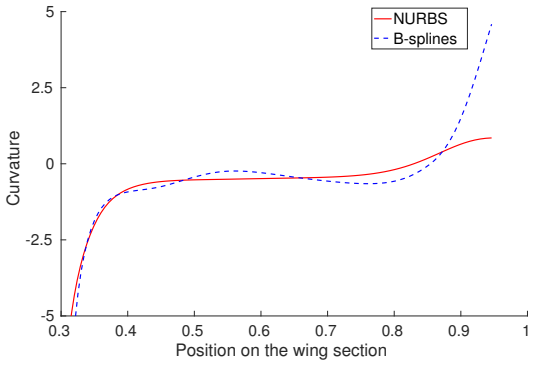
where κ is the curve curvature, primes refer to derivatives w.r.t. the parametric coordinate u in this work. x' and y' are first derivatives, x'' and y'' are second derivatives. It can be seen that the NURBS M6 wing produces smoother wing profiles.



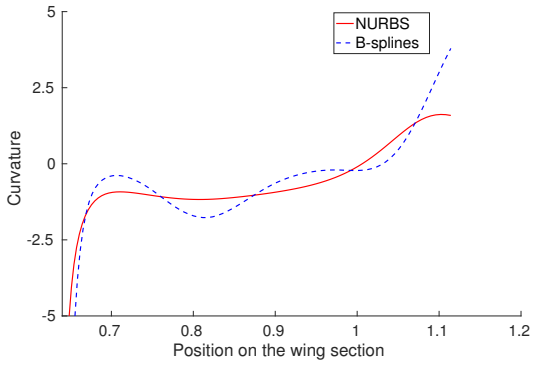
(a) 2.45% span



(b) 16.12% span



(c) 41.48% span



(d) 91.22% span

Figure 9.17: Comparison of curvature along wing sections.

The figures of curve curvature clearly show that the optimisation using a NURBS parametrisation produces a smoother shape. The possible reason is that, when fewer control points are used, the possibility of producing noisy surface is reduced. As a consequence, the aerodynamic performance is better, since in the transonic flow the aerodynamic performance is very sensitive to the shape of geometry. The smoother curvature variation will be more important in viscous flow, as strong changes in curvature will lead to rapid pressure changes which will adversely affect the boundary layer.

The computational time of each part of the optimisation for the NURBS case is given in Table 9.2. In this table, the primal, adjoint, flow sensitivity assembly and mesh deformation are performed within the solver STAMPS. Other parts include optimiser, line search, reading and writing files, etc. It can be observed that STAMPS costs around 97.70% of the whole optimisation run time. Compared to this, the run time of NSPCC and other parts are almost neglectable (around 2.30%). This demonstrates again that the flow and adjoint are most expensive parts in optimisation problem based on CFD and gradient.

Apart from the total run time, the primal and adjoint take 23.14 minutes and 27.38 minutes in the first iteration with the convergence level shown in Fig. 9.18, respectively. As can be seen, the run time of the first iteration is longer than the average runtime for each iteration. This is because that the one-shot method as described by Christakopoulos [61] is utilised, such that only in the first iteration, the flow and adjoint are fully converged. More details on the one-shot method could refer to [175, 176].

Table 9.2: The run time of NURBS M6 wing optimisation for 140 iterations.

Item	Run time (minutes)	Percentage
Adjoint	2932.80	51.13%
Primal	2543.78	44.35%
NSPCC and others	131.68	2.30%
Flow sensitivity assembly	79.55	1.39%
Mesh deformation	47.61	0.83%
Total	5735.42	100%
Average per iteration	40.97	

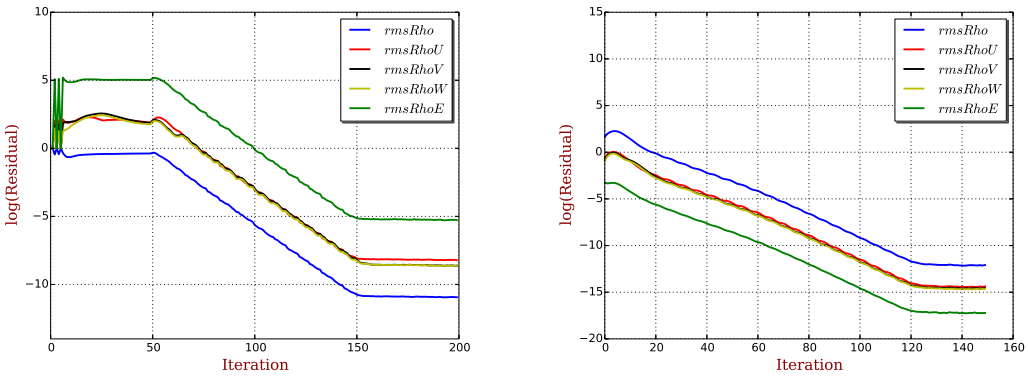


Figure 9.18: Convergence histories of flow (left) and adjoint (right) in the first iteration.

Comparison with published solutions

Agarwal et al. [265] performed a CAD-based and lift-constrained drag minimisation for ONERA M6 wing in the context of Euler flow with the widely used solver SU2. A detailed comparison of parameters between that study and the present work is given in Table 9.3. Note that in [265], the freestream temperature is 288.15K, while in the present work 300K is used. The number of mesh nodes and elements are very similar to those in this study. It would be beneficial to compare the results in the present study with that presented in [265].

Table 9.3: Comparison of parameters between Agarwal's work and the present study.

	Agarwal's work	The present work
Flow type	Euler flow	Euler flow
Freestream temperature	288.15K	300K
Freestream Mach number	0.8395	0.84
Angle of attack	3.06°	3.06°
Objective function	Drag with lift constraint	Drag with lift constraint
Mesh type	therahedral	therahedral
Number of mesh nodes	154,617	135,204
Number of mesh elements	707,115	771,129
Solver	SU2	STAMPS

The comparison of lift coefficient and drag coefficient is given in Table 9.4. Although parameters are slightly different and solvers are not the same, both the lift coefficient and drag coefficient have similar values. This holds not only for initial values, but also after the shapes are optimised. This indicates that the STAMPS solver can simulate the flow reasonably well. In addition, as can be seen from the table, in the present study more drag reduction is achieved, the possible reason is that more design variables are used in the present work. However, the lift coefficient changes larger than that from [265].

Table 9.4: Comparison of results between Agarwal's work and the present study.

	Agarwal's work	The present work
Initial lift coefficient	0.2849	0.28052
Initial drag coefficient	0.011795	0.01281
Optimised lift coefficient	0.2869	0.27308
Optimised drag coefficient	0.010153	0.00828
Drag reduction percentage	14%	36%
Lift change percentage	0.7%	2.66%

The comparison of pressure contour on the upper surface after optimisation is illustrated in Fig. 9.19. As can be seen, in both cases the shock wave has been reduced significantly. Besides, the pressure distribution exhibit similar behaviour, although in the present work the shock has been reduced to a larger extent.

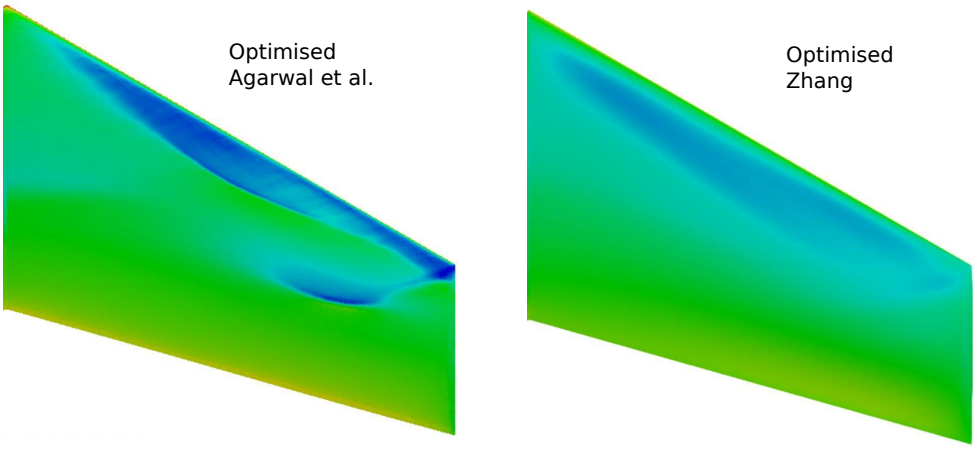


Figure 9.19: Comparison of pressure contours on the upper surface between the literatuire (left) [265] and the present work (right).

Based on the comparison presented above, one can see that the optimised shape obtained in this study is similar to that given by the widely used and famous SU2. This verification indeed gives more confidence to the developed optimisation framework, as well as the STAMPS solver.

9.3.4.3 Optimisation with thickness constraints

From Fig. 9.16 one can observe that in both the B-spline and NURBS cases, the thickness of the wing near the trailing edge is reduced. This indicates that although the lift constraint can present the wing becoming a flat plate when minimising the drag, the wing still become thinner near the trailing edge. A thickness constraint can be imposed to avoid this.

Xu et al. [48] proposed a method to impose thickness constraint using the test point approach. In the present work, a method working directly with the NURBS or B-spline surfaces is developed. The basic idea is to add a penalty term to the cost function, such that a new cost function is obtained. For example, a new cost function is constructed as:

$$J_{new} = \omega_1 J_{STAMPS} + \omega_2 J_{thickness}, \tag{9.8}$$

where J_{STAMPS} is the cost function described in equation (9.2), $J_{thickness}$ is the penalty term for thickness, ω_1 and ω_2 are user supplied weight coefficients which can be changed to specify different importances to the terms.

$J_{thickness}$ is computed at the specified chordwise position, to constraint the vertical separation of the two surfaces. In this study, $J_{thickness}$ is defined as:

$$J_{thickness} = (1 - \frac{T}{T^*})^2, \tag{9.9}$$

where T is the average thickness of wing at the specified chordwise position, and T^* is the initial average thickness. In this way, T^* is setted as the target thickness for the average thickness of the wing at the specified chordwise position. T is defined as:

$$T = \frac{1}{n}\sum_{i=1}^{n}(T_1 + T_2 +, ..., +T_n), \tag{9.10}$$

where n is the number of points in the spanwise direction, T_1, T_2, T_i are the thickness of the wing at these points, as shown in Fig. 9.20. The chordwise position should be specified in the parametric space. The parametric coordinates (u_T in Fig. 9.20) of these points could be obtained using the point inversion algorithm presented in Section 5.3. To use a gradient-based optimiser, the gradient of J_{new} is required, which is

$$\frac{dJ_{new}}{d\boldsymbol{\alpha}} = \omega_1 \frac{dJ_{STAMPS}}{d\boldsymbol{\alpha}} + \omega_2 \frac{dJ_{thickness}}{d\boldsymbol{\alpha}}, \tag{9.11}$$

where $\frac{dJ_{STAMPS}}{d\boldsymbol{\alpha}}$ is simply the sensitivities used in previous optimisations. $\frac{dJ_{thickness}}{d\boldsymbol{\alpha}}$ can be calculated straightforwardly using AD.

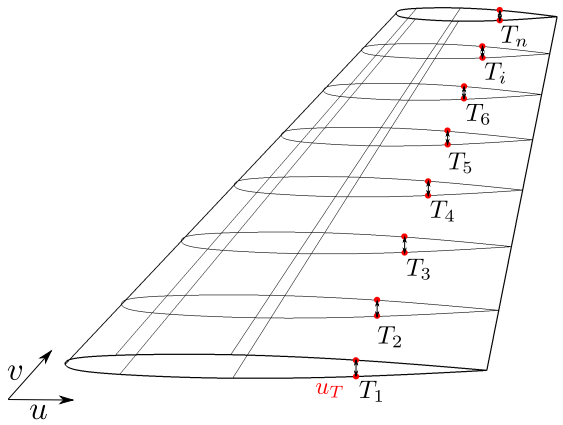


Figure 9.20: Thickness constraints of the ONERA M6 wing

To verify this method, an optimisation is run with thickness constraints for the B-spline M6 wing. 11 points are chosen along the spanwise direction, u_T is 0.9 which is corresponding to 88.2% chord length position, $\omega_1 = 1$, $\omega_2 = 400$. The AD tool Tapenade is employed to compute $\frac{dJ_{thickness}}{d\boldsymbol{\alpha}}$. All other setting parameters and case informations are the same as in the B-spline case presented in Section 9.3.4.2.

The comparison of wing sections at different spanwise locations are shown in Fig. 9.21, illustrating the initial and optimised shapes. As can be seen, when the thickness constraint is imposed, the thickness of the wing at the 88.2% chord length position (the dash line) are larger than the case without thickness constraint, and are just slightly smaller than the initial wing. The value of drag and lift coefficients, as well as the objective function values are listed in Table 9.5. As can be seen, with the thickness constraint imposed, similar aerodynamic performance is obtained while the value of $J_{thickness}$ after optimisation is 8.48×10^{-4}, which means that T is around 97.1% of T^*. This demonstrates the effectiveness of the developed method in maintaining the thickness.

Table 9.5: Values before and after optimisation with thickness constraint.

Case	Drag coefficient	Lift coefficient	J_{STAMPS}	$J_{thickness}$
Initial	0.01281	0.28052	0.01281	0
Without thickness	0.00843	0.27464	0.00859	
With thickness	0.00841	0.27369	0.00859	8.48×10^{-4}

It should be mentioned that the developed method to impose thickness constraint is flexible. Firstly, one can easily set another amount to the target thickness T^* according to design requirement. Secondly, more points could be used along the spanwise direction

at the same chordwise position. Thirdly, one can also impose the thickness constraint at several chordwise positions simultaneously, which can help to achieve the box constraints. These chordwise positions can be easily specified in parametric space after performing the point inversion algorithm. In addition, the definition of the thickness T can also be changed, for example one can use the minimum thickness along the spanwise direction. Finally, the coefficients ω_1 and ω_2 can be modified to achieve different performances. All these modifications can be easily implemented by changing equations (9.9) and (9.10). Then, the forward mode AD can be applied to the modified subroutines to provide $\frac{dJ_{thickness}}{d\boldsymbol{\alpha}}$.

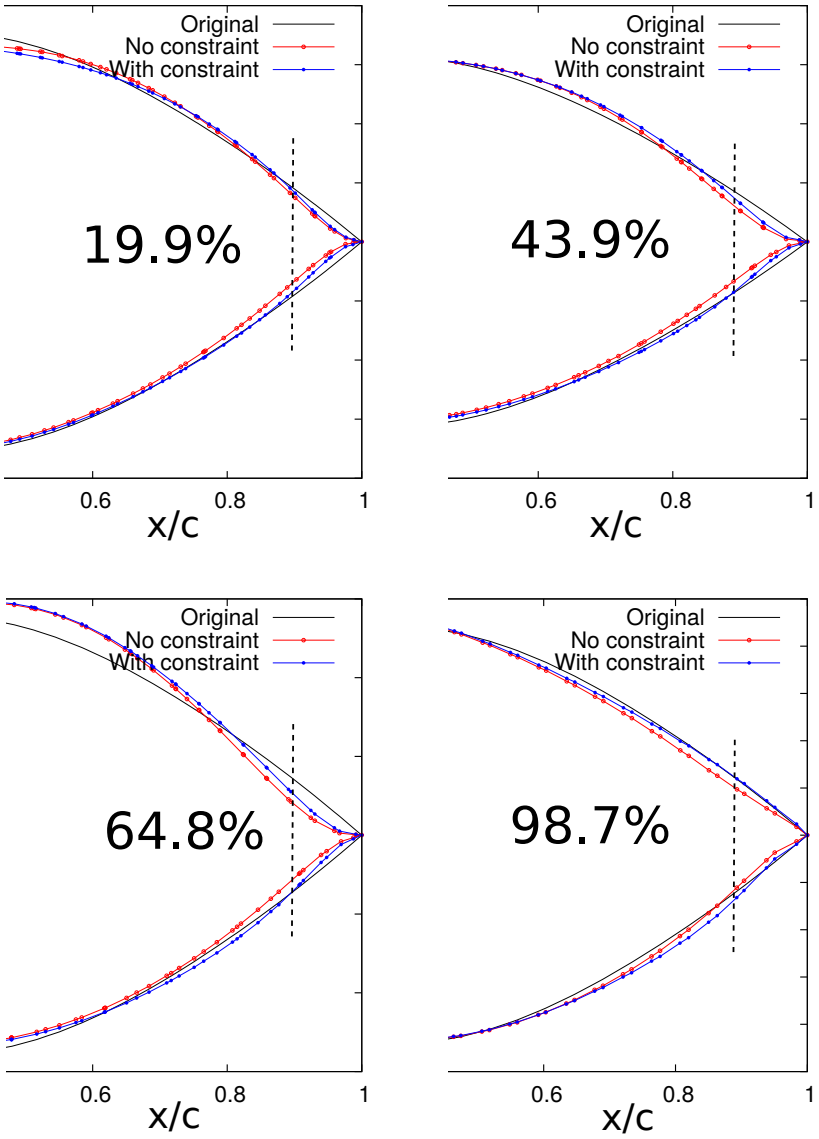


Figure 9.21: Comparison of wing section shapes near the trailing edge at different spanwise locations.

9.4 Summary

In this chapter, the effectiveness of CAD-based optimisation coupling the lightweight NSPCC kernel with adjoint method has been demonstrated with a transonic aerodynamic shape optimisation case.

The ONERA M6 wing is re-parametrised with NURBS, which have been demonstrated as an appropriate parametrisation method for wing shape optimisation. NURBS associate a weight with each control point thus providing additional freedoms in controlling shapes. The optimisation results of ONERA M6 wing with both B-splines and NURBS parametrisation in transonic inviscid flow based on adjoint method have been presented. It has been shown that the optimisation using a NURBS-based wing parametrisation has a smoother shape with smaller variation of curvature, which is beneficial for aerodynamic performance. This can be important in the design of turbo-machinery blades or transonic wings.

A method is developed to impose thickness constraints, working directly with the NUBRS and B-spline surface. This is achieved by adding the thickness constraint as a penalty term to the cost function. The chordwise position can be imposed in the parametric space, and the parametric coordinate can be obtained from the point inversion. The results have shown that the effectiveness of this method in preventing the wing becoming too thin.

Chapter 10

Conclusions and future work

10.1 Summary

With the continuous development of computer technology, CFD-based shape optimisation is increasingly applied in industry. Since CFD evaluations are expensive, the gradient-based optimisation approaches become the method of choice because they can converge in fewer iterations compared to their gradient-free counterparts. The widely application of gradient-based CFD shape optimisation is also promoted by the development of adjoint method, which can provide sensitivities in efficient manner.

Geometry parametrisation directly determines the design space and thus having significant impact on optimisation results in CFD-based shape optimisation problem. There are various kinds of parametrisation methods, which can be broadly categorised as CAD-free and CAD-based ones. The main shortcoming of the CAD-free approaches is that the result does not exist in CAD format. However, in industry, the CAD geometry is highly required since a product design typically starts with a CAD geometry and the optimised geometry should be in CAD format such that it can be manufactured.

This thesis mainly focuses on CAD-based geometry parametrisation and its application in CFD-base shape optimisation problems. The benefits of NURBS in CAD-bsaed shape optimistion, and how to use NURBS effectively is studied. The QMUL in-house CAD kernel NSPCC (NURBS-based parametrisation with continuity constraints) is further investigated, developed and validated.

The underlying principle of NSPCC is further investigated to gain better understanding on the approach. Firstly, the physical meaning of design space when continuity

constraints imposed is made more clear, namely the linear combination of deformation modes, which are basis vectors of the nullspace of the constraint matrix. The linear combinational coefficients are then used as design variables in optimisation. Indeed, due to the presence of the rounding errors in computing, numerical nullspace is utilised instead of theoretical nullspace. Secondly, the deformation modes at different locations of the numerical nullspace have been plotted in Paraview[1] and analysed. The results indicate that these modes have various impacts on the geometry. The influence of number of deformation modes contained in the design space on final results are studied then and it has been found that the number of deformation modes affect the optimisation performance significantly. Thirdly, the factors affecting the required number of test points to maintain continuity constraints are investigated. Results show that the imposed continuity level and the knot vector, as well as weights of control points will change the required numbers.

Based on these investigations and findings, the effective rank is proposed as an automatic pre-conditioner of the numerical nullspace thus the design space. The effective rank avoids the user effort of finding a proper cut-off value by trial and error, hence making the NSPCC kernel more automatic and easier to use. Practical optimisation results have shown the feasibility of the effective rank. In addition, an estimate of the required number of test points is given based on the quantity of control points, degree of curve and knot vector. This allows the user to estimate how many test points should be imposed beforehand, thus saving the set up time significantly.

NSPCC is also further developed in this work to make it more powerful, generic, effective, and user friendly. Firstly, it is extended from B-splines to NURBS. Due to the geometric properties of NURBS, currently NSPCC can support wider range of geometries. Because NURBS are used as the standard of CAD systems, such as CATIA V5, Siemens NX and Rhinoceros, NSPCC now has a better linkage to these systems. Secondly, both position and weight of control points are added into the design space in this study by using the homogeneous form of NURBS. Scaling strategy is utilised to make this works well. Since NURBS provide additional factors controlling the geometry compared to B-splines, i.e. the weights of control points, a geometry can be described using NURBS with fewer control points. This helps to reduce the risk of producing non-smooth surfaces. Thirdly, the capability of imposing constraint is enriched. To be precisely, now different continuity levels can be imposed to different edges as specified by the user, and control points can be fixed in a part of the (x, y, z) directions. These new capabilities of imposing constraints allow NSPCC to handle more geometry optimisation cases. In addition, all programs of NSPCC are currently handled by one single Makefile, and the

[1] https://www.paraview.org/

setting parameters are managed by the JSON format file, which make it much easier to maintain the codes and more user-friendly.

Before performing practical optimisation, the CAD derivatives are verified using several geometries in Chapter 6. The underlying principle of NSPCC is also validated thoroughly through a series of tests. These validation and verification tests make the NSPCC approach more reliable.

An automated CAD-based shape optimisation framework is developed in this study, consists of a flow solver, an adjoint solver, the NSPCC kernel and a gradient-based optimiser. In the framework, the flow solver provides flow solution and the adjoint solver supplies efficient flow sensitivities. The NSPCC is utilised to handle geometry and computed geometric sensitivities. The gradient-based optimiser then selects new values of design variables along the direction pointed by gradient information. The effectiveness of this framework have been demonstrated in three CFD-based shape optimisation cases, coming from the automotive industry, turbo-machine industry and transonic aerodynamics field, respectively. These cases are with varied design complexities and different flow conditions.

It should be mentioned that if the constraint matrix in NSPCC is updated in each design step, the number of deformation modes included in the design space thus the number of design variables may not be a constant. This prevents the coupling of NSPCC with advanced optimisers, such as the BFGS algorithm. In this study, a strategy is proposed to use the BFGS method together with the NSPCC kernel and the feasibility is shown in optimisation cases.

Chapter 7 presents the shape optimisation of an S-bend air duct from the automotive industry, aiming at reducing the pressure drop. This is an incompressible laminar case, the in-house solver GPDE is utilised to provide flow solution and adjoint sensitivity. This case is also devoted to plot deformation modes and to investigate the influence of number of deformation modes on final results. The total pressure loss is reduced significantly by performing a gradient-based optimisation, with the effective rank to determine the design space automatically. G_1 continuity is successfully maintained during the optimisation with the help of the test point-based approach and continuity recovery steps.

In Chapter 8, a U-bend cooling channel from turbo-machinery industry is optimised to reduce the pressure drop. This is a complex turbulent case, the in-house solver STAMPS is employed to analyse the flow and compute flow sensitivities efficiently. In this U-bend case, the geometry is represented with NURBS. In addition, G_0 and G_1 continuity constraints are imposed at different edges between moveable patches and maintained

successfully during the shape optimisation process, which previous version of NSPCC cannot achieve. Apart from the steepest descent method, the BFGS is employed as optimiser coupling NSPCC based on the strategy proposed in this work. The weights of control points are also added into the design space with scaling applied. In all optimisations performed in this chapter, the total pressure loss is significantly reduced and the flow separation present in the datum geometry is considerably eliminated. Among these optimisations, the best results are obtained when the BFGS method is used as optimiser and weights of control points are free.

In Chapter 9, The ONERA M6 wing is optimised in transonic Euler flow condition, aiming at reducing the drag while keeping the lift as a constant. The M6 wing is re-parametrised with NURBS surfaces including weight adjustments, resulting in fewer control points. Both the B-spline and NURBS parametrisations are then utilised in optimisations, where the NSPCC kernel is coupled with the in-house solver STAMPS and a gradient-based optimiser. The results showed that with NURBS parametrisation, lower drag and smoother variation of curvature are achieved, demonstrating the feasibility of NURBS in aerodynamic shape optimisation. In addition, a method is developed to impose thickness constraint to the wing, and the feasibility of this method has been demonstrated by optimisation results.

In this study, geometries of test cases are created with different software, such as CATIA V5, the open-source NURBS toolbox in Matlab[2], Siemens NX 7.5[3] and Rhinoceros V5. This clearly indicates that the advantages of CAD-based parametrisation in exchanging data among different systems by using STEP file. In addition, the developed NSPCC kernel can also be applied in a pure geometric context. For instance, it can be applied to re-parametrise a target geometry, or it can be utilised to perform geometric continuity recovery, as has been demonstrated in this work.

In the present work, NSPCC has been coupled with two different in-house solvers in test cases and works fine, showing the generic property of NSPCC. Although not shown in this work, it is not supervised to predict that NSPCC can also be coupled with other flow and adjoint solvers, and even structural solvers provided the gradients are available. This substantiates the applicability of NSPCC to more optimisation cases.

It should be mentioned that, compared to the start-of-the-art techniques by integrating commercial CAD packages into the optimisation loop, the method developed in the present study can provide accurate and robust CAD sensitivities. What's more, the developed method does not rely heavily on user's experience to define a parametrisation

[2]http://uk.mathworks.com/matlabcentral/fileexchange/26390-nurbs-toolbox-by-d-m-spink
[3]https://www.plm.automation.siemens.com/en/products/nx/

for the geometry, as it is automatic derived from the generic CAD format. Besides, the developed could provide larger design space. However, commercial CAD packages may be more robust and can support more complex geometries.

Finally, it is worthwhile mentioning that some improvements obtained in the performed shape optimisations could possibly be predicted by an experienced designer and fluid mechanics expert, but a human designer would not be in a position to predict the best possible shape. With numerical shape optimisation, the design space can be explored systematically and it is possible to find better shape without heavily relying on experiences.

10.2 Contributions

The main contributions of this thesis are as follows:

1. Investigated the benefits of using NURBS in aerodynamic shape optimisation. Including:
    - Added weights of NURBS control points to design space as additional design freedoms, and proposed a scaling method to use weights effectively
    - Demonstrated that by using NURBS in aerodynamic shape optimisation, better optimisation results can be obtained
    - Successfully computed NURBS derivatives with automatic differentiation (AD) and verified them, thus robust and accurate NURBS derivatives are available
    - Extended the in-house CAD kernel NSPCC from B-splines to NURBS, such that wider range of geometries are supported by the NSPCC CAD kernel
2. Further investigated the underlying principle of the in-house CAD kernel, and developed it to make it more powerful and automatic. This includes:
    - Investigated the constitution of design space and found out how parameters in the CAD kernel affects the performance
    - Proposed a pre-conditioner that can provide a reasonable design space automatically
    - Demonstrated the ability of NSPCC to be coupled with different flow and adjoint solvers

- Enriched the functionalities of the NSPCC kernel, and made it more efficient and easier to use. Reduced its requirement on experience of use

3. Found out the reason why the size of design space changes, and proposed a strategy to couple the NSPCC approach with quasi-Newton method in aerodynamic shape optimisation

4. Developed an automated, modular and user-friendly CAD-based shape optimisation framework. Assessed its performance and demonstrated its effectiveness in various test cases, with different solvers and optimisers utilised

5. Developed a method to impose thickness constraint in wing shape optimisation

6. Performed systematic validation and verification for the NSPCC kernel

A part of findings of this research have been applied to studies performed by the author and the author's colleagues [167, 260, 268–270]. For a full list of publications and conference presentations produced during the present research, please refer to Appendix A.

10.3 Recommendations and future work

Closing this chapter, the author would like to make several recommendations for future research directions based on the conclusions drawn above:

1. Coupling the developed NSPCC method with open source CAD kernel, e.g. OpenCASCADE, to support reading and writing more complex geometries. As stated before, at the moment the geometry reader and writer were developed in the group, which is not as powerful as OpenCASCADE. By coupling NSPCC with those open source CAD kernels, the advantages of them will be fused together, resulting in a more powerful geometric tool and thus a better design and optimisation chain.

2. Coupling NSPCC with more widerly used open source flow solvers, such as OpenFOAM and SU2, to support wider flow situations. During this study, since the aim is to demonstrate the principle of the NSPCC CAD kernel and the effectiveness of the automatic optimisation chain, only in-house solvers were utilised for flow simulation and providing flow sensitivities. Consequently, the test cases employed in this study were relatively simple in terms of flow conditions. It is likely that the in-house solvers are not as robust as widely used open source solvers, for example in-house solvers do not support different kinds of turbulence models. Considering

the generic property of NSPCC, it would be beneficial to integrate more widely used and robust solvers such that the developed CAD kernel and optimisation methods can be applied to wider range of problems.

3. Run optimisation on benchmark test cases, such that it is more convenient to compare results with other groups. For instance, shape optimisation can be applied to cases defined by the AIAA Aerodynamic Design Optimisation Discuss Group (ADODG). This is helpful for solver validation and also the evaluation of optimisation results.

4. Run more grid refinement studies to make sure the method is robust, and can give stable and reasonable results for cases studied. Currently, in the present work, the optimisation cases are used to demonstrate the principle of the developed method, thus not very much attention have been paid to the mesh refinement aspect of the simulation. However, this is important for CFD-based optimisation loop. More work should be done following this direction, i.e. to refine the mesh, and run optimisation again.

5. Applying the developed NSPCC method to more complex problems, after coupling NSPCC with some advanced geometry reader and writer as well as more powerful solvers. For instance, optimisation can be run for unsteady problem, or moving boundary and phase-change problems. Although optimisation have been run for cases with different flow conditions in this work, namely compressible, incompressible, laminar, turbulent, those cases are relative simple. It is worth running the developed method for more complex problems, which are more closely related to real engineering applications. One of the difficulties is that in complex cases, the grid will be complicated, thus may be not easy to handle for flow and adjoint solvers.

6. Calculating intersection curve. This is important in dealing with the shape optimisation problem consists of several intersecting components, like the wing-fuselage configuration. Martin et al. [247, 248] proposed a method to compute the intersection curve based on NURBS derivatives, which are available in the present work. Therefore, computation of intersection curve could be a future working direction based on the derivatives provided by the NSPCC kernel. To achieve this, mesh deformation technique should be utilised to deal with the mesh perturbation at the intersection.

7. Conducting experiments to validate the simulation results. The present work focus on numerical optimisation loop, and have not run experiments to check whether

results are reasonable or not. However, this is of vital importance for CFD simulation results. It would be very helpful to design experiments and conduct validation.

8. Gradually increase the number of design variables during the optimisation, such that a shape optimisation problem can be solved as a sequence of optimisations from the basic parametrisation to refined parametrisations. Some studies [21, 264, 271] have shown that single level parametrisation does not necessarily ensure better design results, even though the design space is richer. Indeed, the NSPCC approach presented in this work has been applied to the U-bend cooling channel optimisation with adaptive parametrisation by Jesudasan and the author [260]. Initial results show that adaptive parametrisation can give better design than the static method, so this is a direction worth further investigation. One way to achieve this is to refine the control points net with the knot insertion algorithm. However, a trade-off should also be made in order to avoid bumpy shapes.

9. Multi-point optimisation. It is more practical to run multi-point shape optimisation for a wing than the single-point one, since in this case the wing will have good performance over a range of operating conditions.

10. Utilising the test point approach to impose continuity constraints between fixed and free surfaces. Currently, this is achieved by freezing a part of control points near the interface, which may reduce the design space to small extent.

11. Applying the NSPCC kernel to structural optimisation or multidisciplinary optimisation. Since NSPCC is solver-neutral, it can both be combined with flow/adjoint solver and structural solver. Therefore, it is possible to perform structural optimisation or multidisciplinary optimisation using NSPCC as the CAD-engine. Multidisciplinary optimisation is also a current working direction of the CFD research group at QMUL[4].

12. Developing a user-friendly GUI. Although the current optimisation framework is automated and easy to use, it still needs some efforts to set up in a non-intuitive manner, such as imposing continuity constraints between patches. Therefore, it would be helpful if a user-friendly GUI can be developed, from which the user can directly click the common edge and simply type in the required continuity level.

[4] http://maddog.sems.qmul.ac.uk/

Appendix A

Author's publications and presentations

During the PhD study, a part of the author's work have been documented in journal and conference proceedings papers, or presented in international conferences, which are listed as following:

Papers

Journal papers

1. **X. Zhang**, Y. Wang, S. Akbarzadeh, J.-D. Müller. Handling geometric continuity constraints with mathematically-derived design variables in CAD-based shape optimisation. *Computer-Aided Design*. To be submitted.

2. S. Akbarzadeh, **X. Zhang**, J.-D. Müller. On the fixed-point discrete adjoint of SIMPLE-type incompressible solvers. To be resubmitted

3. R. Jesudasan, **X. Zhang**, M. Gugala, J.-D. Müller. "CAD-free vs CAD-based parametrisation method in adjoint based aerodynamic shape optimisation". In preparation.

Conference papers

1. **X. Zhang**, Y. Wang, M. Gugala, J.-D. Müller. Geometric continuity constraints for adjacent NURBS patches in shape optimisation. European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS). June 5 - 10, 2016, Crete Island, Greece.

2. **X. Zhang**, R. Jesudasan, J.-D. Müller. Adjoint-based Aerodynamic Optimisation of Wing Shape Using Non-Uniform Rational B-splines. International Conference On Evolutionary and Deterministic Methods For Design Optimisation And Control With Applications to Industrial And Societal Problems (Eurogen 2017), September 13 - 15, 2017 in Madrid, Spain.

3. R. Jesudasan, **X. Zhang**, J.-D. Müller. Adjoint optimisation of internal turbine cooling channel using NURBS-based automatic and adaptive parametrisation method. ASME Gas Turbine Conference. December 7-8, 2017, India.

Conference presentations

Oral presentations

1. **X. Zhang**, J.-D. Müller. "Geometrical continuity constraints for NURBS patches in shape optimisation". Eurogen 2015. September 14-16, 2015, Glasgow, UK.

2. S. Akbarzadeh, Y. Wang, **X. Zhang**, J.-D. Müller. "One-Shot optmisation with fixed-point discrete adjoint of SIMPLE-type incompressible solvers". Eurogen 2015. September 14-16, 2015, Glasgow, UK.

3. J.-D. Müller, S. Xu, M. Gugala, **X. Zhang**, T. Verstraete. "Toward routine use of adjoint design optimization: what works, (and what not quite yet)", keynote speech. 16th International Symposium on Transport Phenomena and Dynamics of Rotating Machinery. April 10-15, Honolulu, Hawaii, USA.

4. R. Jesudasan, **X. Zhang**, M. Gugala, J.-D. Müller. "CAD-free vs CAD-based parametrisation method in adjoint based aerodynamic shape optimisation". European Congress on Computational Methods in Applied Sciences and Engineering. June 5 - 10, 2016, Crete Island, Greece.

5. S. Akbarzadeh, **X. Zhang**, J.-D. Müller. "On the differentiation of SIMPLE algorithm in one-shot methods for optimal design". European Congress on Compu-

tational Methods in Applied Sciences and Engineering. June 5 - 10, 2016, Crete Island, Greece.

6. R. Jesudasan, **X. Zhang**, O. Mykhaskiv, M. Gugala and J.-D. Müller. "Adjoint optimisation of internal turbine cooling channel using node and CAD-based automatic parametrisation methods". European Conference: Simulation-Based Optimisation (NAFEMS-2016). October 12 - 13, 2016, Manchester, UK.

Poster presentation

- S. Akbarzadeh, **X. Zhang**, J.-D. Müller. "One-Shot optmisation with fixed-point discrete adjoint of SIMPLE-type incompressible solvers". Automatic Design Optimisation Seminar. November 23-24, 2015, Derby, UK.

Appendix B

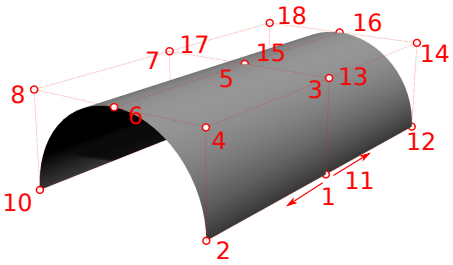
More validation results for NSPCC

In Section 6.3, two tests have been performed to validate the constraint matrix in NSPCC. The perturbations used in those tests were manually chosen to lie in the nullspace. In this appendix, two more tests will be described that the perturbations are not in the nullspace.

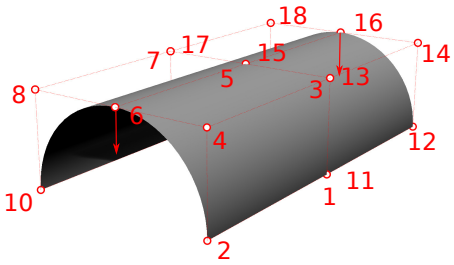
Test 3: Imposing G_0 with $\delta\mathbf{P}$ not in the null space

Same as in Test 1, the G_0 continuity constraint is imposed with 7 test points along the common edge in this test, and all weights are free. The perturbation used in this test is shown in Fig. B.1(a), where the 1st and 11th control point are moved in opposite directions (-y and +y direction, respectively) for the same amount. Obviously, this perturbation will break the G_0 continuity, thus it is not in the nullspace. Therefore, the value of elements in $\mathbf{C}\delta\mathbf{P}$ should not all be 0.

The transpose of $\mathbf{C}\delta\mathbf{P}$ is plotted in Fig. B.2. It can be seen clearly that the value of some elements (the 2nd, 5th and 8th elements) are not 0. From Test 1 and Test 3 one can see that the constraint matrix built in NSPCC can successfully figure out whether the perturbations belong to the nullspace or not. In other words, the constraint matrix for G_0 continuity is reliable.



(a) Perturbation of Test 3: not in the nullspace of G_0 continuity



(b) Perturbation of Test 4: not in the nullspace of G_1 continuity

Figure B.1: The half-cylinder geometry and perturbations for Test 3 and Test 4.

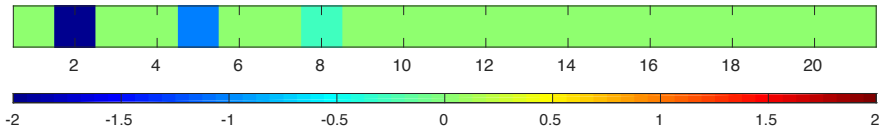


Figure B.2: The transpose of $\mathbf{C}\delta\mathbf{P}$ in Test 3.

Test 4: Imposing G_1 with $\delta\mathbf{P}$ not in the null space

Same as in Test 2, the G_1 continuity constraint is imposed with 9 test points along the common edge in this test. But in this test both the 6th and 16th control point are moved towards -z direction for the same amount, as shown in Fig. B.1(b). In this case, the 5th, 6th, 15th and 16th control point will no longer lie in one straight line after perturbation, so the G_1 continuity will be broken. Therefore, the perturbations are not in the nullspace, and the value of $\mathbf{C}\delta\mathbf{P}$ should not be 0.

The transpose of $\mathbf{C}\delta\mathbf{P}$ is plotted in Fig. B.3, which illustrates that some elements in $\mathbf{C}\delta\mathbf{P}$ are not 0. From Test 2 and Test 4, one can see the constraint matrix for G_1 continuity is reliable.

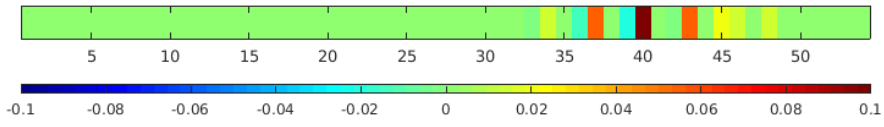


Figure B.3: The transpose of $\mathbf{C}\delta\mathbf{P}$ in Test 4.

Appendix C

LAPCAK

LAPACK[1] is an open source library of programs that used to solve numerical linear algebra problems [227–229]. In LAPACK, the driver routines are used for solving complete problems like singular value problems, linear equations, linear least squares problems. The computation routines are designed for performing distinct computational task. Each driver routine calls a sequence of computational routines.

In the NSPCC kernel, the Singular Value Decomposition (SVD) is essential, because it gives the singular values and nullspace of the constraint matrix, which are utilised to impose geometric continuity. In addition, SVD is also used to compute the pseudo-inverse, which is needed in continuity recovery steps. Therefore, LAPACK is employed because it provides subroutines to perform SVD.

Actually, LAPACK provides several subroutines to conduct SVD using different methods. The subroutine utilised in the NSPCC kernel to perform SVD in equation (5.18) is DGESDD()[2], which is called as:

```
1  CALL DGESDD('A', M, N, C_matrix_input, LDA, SV, Umatrix, LDU,
2  & VTmatrix, LDVT, WORK, LWORK, IWORK, INFO2)
```

Arguments are:

- 'A'. This parameter means that all eigenvectors will be returned in the array Umatrix and VTmatrix. This is crucial, since the nullspace is obtained from VTmatrix.
- M. The number of rows of the constraint matrix **C**.

[1] http://www.netlib.org/lapack/
[2] http://www.netlib.org/lapack/lapack-3.1.1/html/dgesdd.f.html

- N. The number of columns of **C**.
- C_matrix_input. The constraint matrix **C**.
- LDA. The leading dimension of **C**.
- SV. The singular values of **C**.
- Umatrix. The matrix **K** in equation (5.18).
- LDU. The leading dimension of **K**.
- VTmatrix. The matrix $\mathbf{V}^T$ in equation (5.18).
- LDVT. The leading dimension of $\mathbf{V}^T$.
- WORK. If INFO2 = 0, WORK(1) returns the optimal LWORK.
- LWORK. The dimension of array WORK.
- IWORK. Dimension (8×min(M, N))
- INFO2. This is an integer used to indicate whether the routine runs successfully. 'INFO2 = 0' means successful. 'INFO2 = -i' means the i-th argument has an illegal value. 'INFO2 > 0' means the updating process failed.

LAPACK is easy to install, configure and use. More information can be found on the LAPACK website[3].

[3] http://www.netlib.org/lapack/

Appendix D

AD tool Tapenade and example

In this work, the AD tool Tapenade[1] is frequently used to compute accurate derivatives.

Firstly, even though the development of solvers is not the focus of this work, Tapenade is used extensively in deriving the discrete adjoint solver, since the hand implementation of the discrete adjoint solver is tedious and error-prone. The interested reader on the implementation details of our in-house discrete adjoint solvers is referred to my colleagues' work [167, 180].

Secondly, in computing the NURBS derivatives (see Section 6.1.1) and the derivatives of continuity cost function with respect to scaled control points, AD are also utilised. Since NSPCC is written in Fortran 90, thus Tapenade is chosen.

Tapenade is an Automatic Differentiation Engine developed at INRIA Sophia-Antipolis by the Tropics then Ecuador teams[2]. It is based on the source code transformation (S-T) (see Section 3.4.4) and can recognise Fortran 90. In this work, Tapenade is used to generate the differentiated codes which will then be integrated into the NSPCC framework manually. Generally speaking, Tapenade can be used as a black-box tool, provided that the source program is given and inputs as well as outputs are identified. Also, the forward mode or reverse mode has to be specified.

Tapenade can be used in several ways. Firstly, it can be used in command line or in Makefile. Secondly, there is an online AD engine[3] which can be easily run. Finally, Tapenade also has a graphic user interface (GUI) for using it conveniently, as shown in Fig. D.1. As can be seen, the input language, source file, subroutine to be differentiated,

[1] http://www.autodiff.org/?module=Tools&tool=TAPENADE
[2] http://www-sop.inria.fr/tropics/tapenade.html
[3] http://www-tapenade.inria.fr:8080/tapenade/

input and output variables, as well we the differentiation mode can all be specified in this GUI.

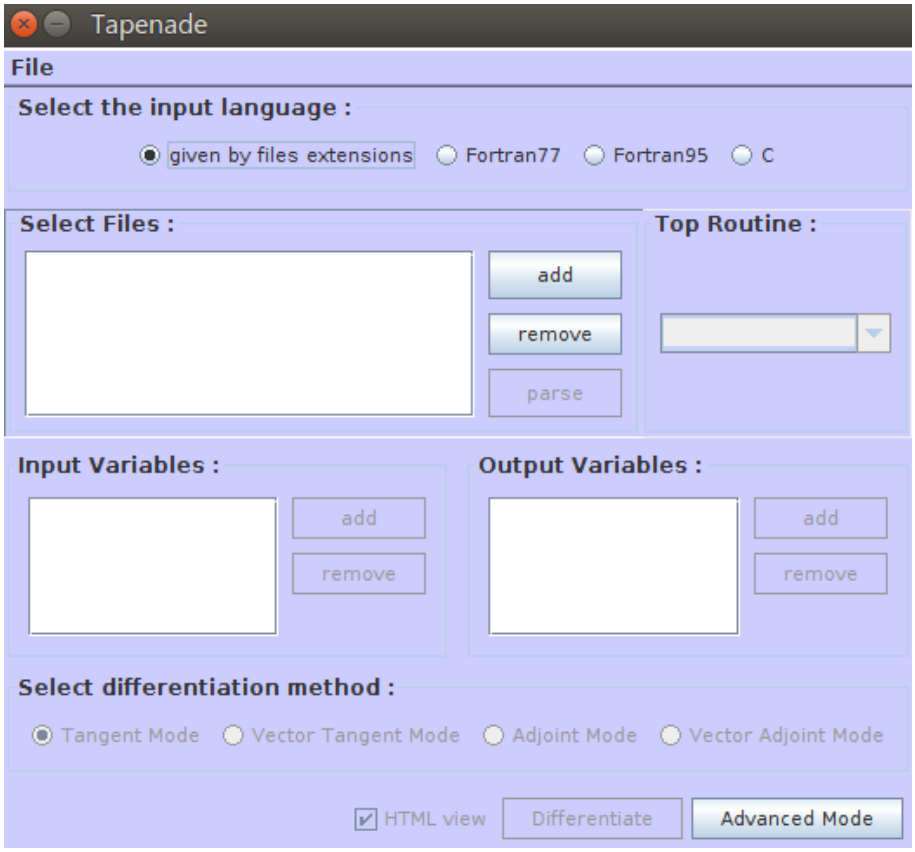


Figure D.1: The graphic user interface of Tapenade.

In the author's opinion, even though it takes sometime to learn AD and Tapenade, it is really worthy. AD and Tapenage will make the development process more conveniently and quicker.

Example of using Tapenade

Computing the NURBS derivatives

As presented in Section 6.1.1, Tapenade is used in forward mode to compute the NURBS derivatives. The source codes calling differentiated subroutine to obtain derivatives are listed below.

```
1     ! dS/du
```

```
2    ud=1.0; vd=0.0; cp_w_d = 0
3    call S_d(u, ud, v, vd, cp_w, cp_w_d, s, sd)
4    dSdu=sd
5
6    ! dS/dv
7    ud=0.0; vd=1.0; cp_w_d = 0
8    call S_d(u, ud, v, vd, cp_w, cp_w_d, s, sd)
9    dSdv=sd
10
11   ! dS/d(wx)
12   ud=0.0; vd=0.0; cp_w_d=0.0; cp_w_d(1)=1.0
13   call S_d(u, ud, v, vd, cp_w, cp_w_d, s, sd)
14   dSdwx=sd
15
16   ! dS/d(wy)
17   ud=0.0; vd=0.0; cp_w_d=0.0; cp_w_d(2)=1.0
18   call S_d(u, ud, v, vd, cp_w, cp_w_d, s, sd)
19   dSdwy=sd
20
21   ! dS/d(wz)
22   ud=0.0; vd=0.0; cp_w_d=0.0; cp_w_d(3)=1.0
23   call S_d(u, ud, v, vd, cp_w, cp_w_d, s, sd)
24   dSdwz=sd
25
26   ! dS/dw
27   ud=0.0; vd=0.0; cp_w_d=0.0; cp_w_d(4)=1.0
28   call S_d(u, ud, v, vd, cp_w, cp_w_d, s, sd)
29   dSdw=sd
```

Listing D.1: Computing first NURBS derivatives using differentiated subroutine.

```
1    ! dsds/dudu
2    ud=1.0; udd=1.0;
3    vd=0.0; vdd=0.0;
4    call S_d_d(u, ud, udd, v, vd, vdd, cp_w, s, sd, sdd)
5    dsds/dudu = sdd
6
7    ! dsds/dvdv
8    ud=0.0; udd=0.0;
9    vd=1.0; vdd=1.0;
10   call S_d_d(u, ud, udd, v, vd, vdd, cp_w, s, sd, sdd)
11   dsds/dvdv = sdd
12
13   ! dsds/dudv
14   ud=1.0; udd=0.0;
15   vd=0.0; vdd=1.0;
```

```
16    call S_d_d(u, ud, udd, v, vd, vdd, cp_w, s, sd, sdd)
17    dsds/dudv = sdd
```

Listing D.2: Computing second NURBS derivatives using differentiated subroutine.

Another example

Another example of using AD and Tapenade in this work is the differentiation of the cost function in equation (9.1). The original subroutine computing the cost function is:

```
1   ! compute the cost function
2   subroutine calc_J(Xs_initial, Xs_target, J)
3      implicit none
4      real(8), dimension(:,:), allocatable            :: Xs_target
5      real(8), dimension(:,:), allocatable            :: Xs_initial
6      real(8) :: J
7      integer :: i, number_point
8
9      J=0.0
10     number_point = size(Xs_initial,1)
11     do i=1, number_point
12         J = J + sum((Xs_initial(i,:)-Xs_target(i,:))**2)
13     enddo
14     J=J/number_point
15     J=sqrt(J)
16  end subroutine calc_J
```

The differentiated subroutine using AD is:

```
1   !        Generated by TAPENADE     (INRIA, Ecuador team)
2   !  Tapenade 3.12 (r6213) - 13 Oct 2016 10:54
3   !
4   !  Differentiation of calc_j in forward (tangent) mode:
5   !   variations   of useful results: j
6   !   with respect to varying inputs: *xs_initial
7   !   RW status of diff variables: *xs_initial:in j:out
8   !   Plus diff mem management of: xs_initial:in
9   ! compute the cost function
10  SUBROUTINE CALC_J_D(xs_initial, xs_initiald, xs_target, j, jd)
11    IMPLICIT NONE
12    REAL*8, DIMENSION(:, :), ALLOCATABLE :: xs_target
13    REAL*8, DIMENSION(:, :), ALLOCATABLE :: xs_initial
14    REAL*8, DIMENSION(:, :), ALLOCATABLE :: xs_initiald
15    REAL*8 :: j
```

```
16      REAL*8 :: jd
17      INTEGER :: i, number_point
18      INTRINSIC SIZE
19      INTRINSIC SUM
20      INTRINSIC SQRT
21      REAL*8, DIMENSION(3) :: arg1
22      REAL*8, DIMENSION(3) :: arg1d
23      j = 0.0
24      number_point = SIZE(xs_initial, 1)
25      jd = 0.0_8
26      DO i=1,number_point
27        arg1d(:) = 2*(xs_initial(i, :)-xs_target(i, :))*xs_initiald(i, :)
28        arg1(:) = (xs_initial(i, :)-xs_target(i, :))**2
29        jd = jd + SUM(arg1d(:))
30        j = j + SUM(arg1(:))
31      END DO
32      jd = jd/number_point
33      j = j/number_point
34      IF (j .EQ. 0.0) THEN
35        jd = 0.0_8
36      ELSE
37        jd = jd/(2.0*SQRT(j))
38      END IF
39      j = SQRT(j)
40    END SUBROUTINE CALC_J_D
```

Appendix E

JSON format

In both the NSPCC approach and the CAD-based shape optimisation framework, some input parameters should be given so that the program can run smoothly and successfully.

Previously, these input parameters are given in a messy way, i.e. directly set in codes while programming. This is very inconvenient, because each time when a parameter needs to be changed, it should be modified in the codes and then one should recompile the program. Therefore, one task in this project is to find a better way to set input parameters.

The JavaScript Object Notation (JSON) format [272, 273] is chosen in this work. It is an open-standard form that uses human-readable text to transmit data objects consisting of attribute-value pairs[1, 2]. This is easy for both the human and computer to read. By using the JSON format, there is no need to recompile the program if users only want to change the input parameters but keep the program itself untouched.

The following is a part of the JSON file used in NSPCC, which is corresponding to the FINDPARAMS module. As can be seen, it is very easy to understand.

```
"findParams":
        {
        "inputFiles":
            {
            "surfaceInputFile":"B-Spline-Surface-input",
            "coordinatesOfMesh" :"../testcase/ubend/
```

[1] https://en.wikipedia.org/wiki/JSON
[2] http://www.json.org/

```
                surf_coords.dat"
        },
    "outputFiles":
        {
        "coordinates&distance":"FOUND",
        "parametricCoordinates" :"uv_params"
                    },
    "pointInversion":
        {
         "searchGridDimension":20,
         "maxIterationNewton" :10,
         "tolerance1"         :0.0001,
         "tolerance2"         :1.0
        }
    },
```

Appendix F

Governing equations in STAMPS

Since the flow solver is not the focus of this research, some detailed expression of the flow equations in STAMPS are put here to make the thesis more compact.

In equation (4.7), the conservative variables vector $\mathbf{W}$ is:

$$\mathbf{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \\ \hat{\nu} \end{bmatrix} \tag{F.1}$$

where ρ is density, ρu, ρv, ρw are momentum in x, y, z direction, respectively. ρe is energy, $\hat{\nu}$ is the Spalart-Allmaras variable.

The convective flux vector $\mathbf{f}_c$ is defined in equation (F.2):

$$\mathbf{f}_c = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho e V + p V \\ \hat{\nu} V \end{bmatrix} \tag{F.2}$$

where p is pressure, u, v, w are velocity components of the velocity vector $\mathbf{u}_v$, n_x, n_y, n_z are the components of the surface unit normal vector $\mathbf{n}$. V is the inner product of $\mathbf{u}_v$ and $\mathbf{n}$, i.e. $V = \mathbf{u}_v \cdot \mathbf{n}$.

The viscous flux vector $\mathbf{f}_v$ is given in equation (F.3)

$$\mathbf{f}_c = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_z\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_z\tau_{yy} + n_z\tau_{yz} \\ n_x\tau_{zx} + n_z\tau_{zy} + n_z\tau_{zz} \\ n_x\Theta_x + n_y\Theta_y + n_z\Theta_z \\ \frac{1}{\sigma}(\nu_L + \hat{\nu})(\nabla\nu \cdot \vec{n})\hat{\nu} \end{bmatrix} \tag{F.3}$$

where τ_{ij} are stresses, Θ_i are are terms describing the work of the viscous stresses and of the heat conduction in the fluid, respectively, as follows:

$$\begin{aligned} \Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz}k\frac{\partial T}{\partial x} \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz}k\frac{\partial T}{\partial y} \\ \Theta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz}k\frac{\partial T}{\partial z} \end{aligned} \tag{F.4}$$

Currently in STAMPS, the heat conduction is not considered.

The source term $\mathbf{q}$ is defined in equation (F.5):

$$\mathbf{q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ SA_{src} \end{bmatrix} \tag{F.5}$$

where SA_{src} is the Spalart-Allmaras source term. As you can see, this term has all 0 components except for the last one.

Bibliography

[1] A. Bentamy, F. Guibault, and J. Trepanier. Aerodynamic optimization of a realistic aircraft wing. In *43rd AIAA Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings*, 2005.

[2] N. Kroll, N. R. Gauger, J. Brezillon, R. Dwight, A. Fazzolari, D. Vollmer, K. Becker, H. Barnewitz, V. Schulz, and S. Hazra. Flow simulation and shape optimization for aircraft design. *Journal of Computational and Applied Mathematics*, 203(2):397 – 411, 2007.

[3] S. Kammerer, J. Mayer, M. Paffrath, U. Wever, and A. Jung. Three-dimensional optimization of turbomachinery bladings using sensitivity analysis. In *Proceedings of ASME Turbo expo 2003, Power for Land, Sea and Air*, June 16-19, Atlanta Georgia, USA, GT2003-38037, 2003.

[4] H.-D. Li, L. He, Y. Li, and R. Wells. Blading aerodynamics design optimization with mechanical and aeromechanical constraints. In *ASME Turbo Expo 2006: Power for Land, Sea, and Air*, pages 1319–1328, 2006.

[5] F. Muyl, L. Dumas, and V. Herbert. Hybrid method for aerodynamic shape optimization in automotive industry. *Computers & Fluids*, 33(5):849–858, 2004.

[6] W.-H. Hucho. *Aerodynamics of road vehicles: from fluid mechanics to vehicle engineering*. Elsevier, 2013.

[7] C. Othmer. Adjoint methods for car aerodynamics. *Journal of Mathematics in Industry*, 4(1):1–23, 2014.

[8] C. Othmer and T. Grahs. Approaches to fluid dynamic optimization in the car development process. In *International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control With Applications to Industrial and Societal Problems*, 2005.

[9] C. Othmer. A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *International Journal for Numerical Methods in Fluids*, 58(8):861–877, 2008.

[10] R. M. Hicks, E. M. Murman, and G. N. Vanderplaats. An assessment of airfoil design by numerical optimization. *NASA TM X-3092*, Ames Research Center, Moffett Field, California, July 1974.

[11] R. M. Hicks and P. A. Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412, 1978.

[12] C.-Y. Joh. Development of a NURBS-based wing design optimization system. In *7th Korea-Russia International Symposium on Science and Technology, Proceedings KORUS 2003*, volume 1, pages 249–254, July 2003.

[13] H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[14] J.-D. Müller. *Essentials of Computational Fluid Dynamics*. Taylor & Francis, 2015.

[15] J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer Science & Business Media, 2012.

[16] A. Jameson and K. Ou. Optimization methods in computational fluid dynamics. *Encyclopedia of Aerospace Engineering*, 36(9):1–14, 2010.

[17] B. Epstein, S. Peigin, and S. Tsach. A new efficient technology of aerodynamic design based on CFD driven optimization. *Aerospace science and technology*, 10(2):100–110, 2006.

[18] D. Thévenin and G. Janiga. *Optimization and Computational Fluid Dynamics*. Optimization and Computational Fluid Dynamics. Springer Berlin Heidelberg, 2008.

[19] S. Amaral, T. Verstraete, R. Van den Braembussche, and T. Arts. Design and optimization of the internal cooling channels of a high pressure turbine blade—part i: Methodology. *Journal of Turbomachinery*, 132(2):021013, 2010.

[20] M. J. Garcia, P. Boulanger, and S. Giraldo. CFD based wing shape optimization through gradient-based method. In *Proceedings of the International Conference on Engineering Optimization. Rio de Janeiro, Brazil*, 2008.

[21] N. Kumar, A. Diwakar, S. K. Attree, and S. Mittal. A method to carry out shape optimization with a large number of design variables. *International Journal for Numerical Methods in Fluids*, 71(12):1494–1508, 2013.

[22] J. R. R. A. Martins. Wing design via numerical optimization. *SIAG/OPT Views-and-News*, 23(2):2–7, April 2015.

[23] M. Jureczko, M. Pawlak, and A. Mężyk. Optimisation of wind turbine blades. *Journal of materials processing technology*, 167(2):463–471, 2005.

[24] X. Wang, W. Z. Shen, W. J. Zhu, J. N. Sørensen, and J. Chen. Shape optimization of wind turbine blades. *Wind Energy*, 12(8):781–803, 2009.

[25] S. Willeke and T. Verstraete. Adjoint optimization of an internal cooling channel U-bend. In *ASME Turbo Expo 2015: Turbine Technical Conference and Exposition*, GT2015-43423, 2015.

[26] T. Verstraete, F. Coletti, J. Bulle, T. Vanderwielen, and T. Arts. Optimization of a U-bend for minimal pressure loss in internal cooling channels Part I: Numerical method. *Journal of Turbomachinery*, 135(5):051015, 2013.

[27] F. Coletti, T. Verstraete, J. Bulle, T. Van der Wielen, N. Van den Berge, and T. Arts. Optimization of a U-bend for minimal pressure loss in internal cooling channels - Part II: Experimental validation. *Journal of Turbomachinery*, 135(5):051016, 2013.

[28] L. Espath, R. Linn, and A. Awruch. Shape optimization of shell structures based on NURBS description using automatic differentiation. *International Journal for Numerical Methods in Engineering*, 2011.

[29] L. Espath, A. Braun, A. Awruch, and S. Maghous. NURBS-based three-dimensional analysis of geometrically nonlinear elastic structures. *European Journal of Mechanics-A/Solids*, 47:373–390, 2014.

[30] R. Linn, L. Espath, and A. Awruch. Optimal shape of axisymmetric solids using NURBS and automatic differentiation. *Applied Mathematical Modelling*, 38(4):1385–1402, 2014.

[31] D. W. Zingg, M. Nemec, and T. H. Pulliam. A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, 17(1-2):103–126, 2008.

[32] A. Taherkhani, G. de Boer, P. Gaskell, C. Gilkeson, R. Hewson, A. Keech, H. Thompson, and V. Toropov. Aerodynamic drag reduction of emergency response vehicles. *Advances in Automobile Engineering*, 4(2), 2015.

[33] A. R. Taherkhani, C. Gilkeson, P. Gaskell, R. Hewson, V. Toropov, A. Rezaienia, and H. Thompson. Aerodynamic cfd based optimization of police car using Bezier curves. *SAE International Journal of Materials and Manufacturing*, 10(2):2017–01–9450, 2017.

[34] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM-Review*, 10(1):110–112, 1998.

[35] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[36] U. Naumann. *The art of differentiating computer programs: an introduction to algorithmic differentiation*. SIAM, 2011.

[37] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(1):97–110, 006 1974.

[38] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.

[39] A. Jameson. Optimum aerodynamic design using CFD and control theory. *AIAA paper*, 1729:124–131, 1995.

[40] M. B. Giles, M. C. Duta, J.-D. Müller, and N. A. Pierce. Algorithm developments for discrete adjoint methods. *AIAA journal*, 41(2):198–205, 2003.

[41] D. Jones, J.-D. Müller, and F. Christakopoulos. Preparation and assembly of discrete adjoint CFD codes. *Computers & Fluids*, 46(1):282–286, 2011.

[42] F. Christakopoulos, D. Jones, and J.-D. Müller. Pseudo-timestepping and verification for automatic differentiation derived CFD codes. *Computers & Fluids*, 46(1):174–179, 2011.

[43] J. Samareh. Aerodynamic shape optimization based on free-form deformation. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Reston, Virigina, Aug 2004.

[44] M. Hojjat, E. Stavropoulou, and K.-U. Bletzinger. The vertex morphing method for node-based shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 268:494 – 513, 2014.

[45] S. Auriemma, M. Banovic, O. Mykhaskiv, H. Legrand, J.-D. Müller, and A. Walther. Optimisation of a U-bend using CAD-based adjoint method with differentiated CAD kernel. In *ECCOMAS Congress*, 2016.

[46] T. Verstraete, L. Müller, and J.-D. Müller. Adjoint-based design optimisation of an internal cooling channel U-bend for minimised pressure losses. *International Journal of Turbomachinery, Propulsion and Power*, 2(2):10, 2017.

[47] G. Yu, J.-D. Müller, D. Jones, and F. Christakopoulos. CAD-based shape optimisation using adjoint sensitivities. *Computers & Fluids*, 46(1):512–516, 2011.

[48] S. Xu, W. Jahn, and J.-D. Müller. CAD-based shape optimisation with CFD using a discrete adjoint. *International Journal for Numerical Methods in Fluids*, 74(3):153–168, 2014.

[49] S. Xu, D. Radford, M. Meyer, and J.-D. Müller. CAD-based adjoint shape optimisation of a one-stage turbine with geometric constraints. In *Proceedings of ASME Turbo Expo 2015: Turbine Technical Conference and Exposition*, 2015.

[50] M. Martín, E. Andrés, M. Widhalm, P. Bitrián, and C. Lozano. Non-uniform rational B-splines-based aerodynamic shape design optimization with the DLR TAU code. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 226(10):1225–1242, 2012.

[51] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 2nd edition, 1997.

[52] D. F. Rogers. *An introduction to NURBS: with historical perspective*. Elsevier, 2000.

[53] J. A. Samareh. Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization. *AIAA journal*, 39(5):877–884, 2001.

[54] D. A. Field. Education and training for CAD in the auto industry. *Computer-Aided Design*, 36(14):1431–1437, 2004.

[55] K. Konno, Y. Tokuyama, and H. Chiyokura. A G^1 connection around complicated curve meshes using C^1 NURBS boundary gregory patches. *Computer-Aided Design*, 33(4):293–306, 2001.

[56] J. Zheng, G. Wang, and Y. Liang. GC^n continuity conditions for adjacent rational parametric surfaces. *Computer Aided Geometric Design*, 12(2):111–129, 1995.

[57] H. Gagnon and D. W. Zingg. Geometry generation of complex unconventional aircraft with application to high-fidelity aerodynamic shape optimization. In *21st AIAA Computational Fluid Dynamics Conference*, page 2850, 2013.

[58] J. T. Hwang and J. R. R. A. Martins. GeoMACH: geometry-centric MDAO of aircraft configurations with high fidelity. In *Proceedings of the 14th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, Indianapolis, Indiana*, 2012.

[59] A. Bentamy, J.-Y. Trépanier, and F. Guibault. Wing shape optimization using a constrained NURBS surface geometrical representation. In *ICAS Congress*, 2002.

[60] L. Piegl. On NURBS: a survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, 1991.

[61] F. Christakopoulos. *Sensitivity computation and shape optimisation in aerodynamics using the adjoint methodology and Automatic Differentiation.* PhD thesis, Queen Mary University of London, September 2012.

[62] A. Jameson and J. C. Vassberg. Studies of alternative numerical optimization methods applied to the brachistochrone problem. *Computational Fluid Dynamics Journal*, 9(3):281–296, 2000.

[63] A. Jaworski and J.-D. Müller. Toward modular multigrid design optimisation. In *Advances in Automatic Differentiation*, pages 281–291. Springer, 2008.

[64] G. Becker, M. Schäfer, and A. Jameson. An advanced NURBS fitting procedure for post-processing of grid-based shape optimizations. In *49th AIAA Aerospace Sciences Meeting, Orlando, FL, USA*, 2011.

[65] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics*, 20(4):151–160, 1986.

[66] E. I. Amoiralis and I. K. Nikolos. Freeform deformation versus B-spline representation in inverse airfoil design. *Journal of Computing and Information Science in Engineering*, 8(2):024001, 2008.

[67] R. Duvigneau. *Adaptive parameterization using free-form deformation for aerodynamic shape optimization.* PhD thesis, INRIA, 2006.

[68] C. Lee, D. Koo, and D. W. Zingg. Comparison of B-spline surface and free-form deformation geometry control for aerodynamic optimization. *AIAA Journal*, 2016.

[69] E. Andrés, P. Bitrián, M. Martin, and M. Widhalm. Preliminary comparison between two CAD-based aerodynamic shape optimization approaches using adjoint methods for fast gradient computation. In *2nd International Conference on Engineering Optimization, Lisbon, Portugal*, 2010.

[70] H. Sobieczky. Parametric airfoils and wings. In *Recent Development of Aerodynamic Design Methodologies*, pages 71–87. Springer, 1999.

[71] D. J. Poole, C. B. Allen, and T. C. S. Rendall. Aerofoil design variable extraction for aerodynamic optimization. In *21st AIAA Computational Fluid Dynamics Conference, San Diego, California*, 2013.

[72] D. A. Masters, N. J. Taylor, T. Rendall, C. B. Allen, and D. J. Poole. Review of aerofoil parameterisation methods for aerodynamic shape optimisation. In *53rd AIAA Aerospace Sciences Meeting*, Reston, Virginia, Jan 2015. American Institute of Aeronautics and Astronautics.

[73] D. A. Masters, N. J. Taylor, T. Rendall, C. B. Allen, and D. J. Poole. A geometric comparison of aerofoil shape parameterisation methods. Number January, pages 1–35, 2016.

[74] D. A. Masters, D. J. Poole, N. J. Taylor, T. Rendall, and C. B. Allen. Impact of shape parameterisation on aerodynamic optimisation of benchmark problem. *54th AIAA Aerospace Sciences Meeting*, (January):1–14, 2016.

[75] D. J. Poole, C. B. Allen, and T. C. S. Rendall. Control point-based aerodynamic shape optimization applied to AIAA ADODG test cases. In *53rd AIAA Aerospace Sciences Meeting*, page 1947, 2015.

[76] D. J. Poole, C. B. Allen, and T. C. S. Rendall. Metric-based mathematical derivation of efficient airfoil design variables. *AIAA Journal*, 53(5):1349–1361, 2015.

[77] C. B. Allen, D. J. Poole, and T. Rendall. Efficient modal design variables applied to aerodynamic optimization of a modern transport wing. In *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 3215, 2016.

[78] D. J. Poole, C. B. Allen, and T. C. S. Rendall. High-fidelity aerodynamic shape optimization using efficient orthogonal modal design variables with a constrained global optimizer. *Computers & Fluids*, 143:1 – 15, 2017.

[79] C. B. Allen, D. J. Poole, and T. C. S. Rendall. Wing aerodynamic optimization using efficient mathematically-extracted modal design variables. *Optimization and Engineeing*, 2018.

[80] K. Yonekura and O. Watanabe. A shape parameterization method using principal component analysis in applications to parametric shape optimization. *Journal of Mechanical Design*, 136(12):121401, 2014.

[81] D. J. J. Toal, N. W. Bressloff, A. J. Keane, and C. M. E. Holden. Geometric filtration using proper orthogonal decomposition for aerodynamic design optimization. *AIAA Journal*, 48(5):916–928, 2010.

[82] S. S. Ghoman, Z. Wang, P. Chen, and R. K. Kapania. A POD-based reduced order design scheme for shape optimization of air vehicles. *AIAA Paper*, 1808, 2012.

[83] H.-Y. Wu, S. Yang, F. Liu, and H.-M. Tsai. Comparison of three geometric representations of airfoils for aerodynamic optimization. 2003.

[84] W. Song and A. J. Keane. A study of shape parameterisation methods for airfoil optimisation. In *Proceedings of the Tenth AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004.

[85] A. Morris, C. Allen, and T. Rendall. CFD-based optimization of aerofoils using radial basis functions for domain element parameterization and mesh deformation. *International Journal for Numerical Methods in Fluids*, 58(8):827–860, 2008.

[86] J. Gräsel, A. Keskin, M. Swoboda, H. Przewozny, and A. Saxer. A full parametric model for turbomachinery blade design and optimisation. In *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 907–914, 2004.

[87] T. Verstraete. CADO: a computer aided design and optimization tool for turbomachinery applications. *International Conference on Enginnering Optimization, Lisbon 2010*, pages 1–10, 2010.

[88] I. S. Torreguitart, T. Verstraete, and L. Mueller. CAD kernel and grid generation algorithmic differentiation for turbomachinery adjoint optimization. In *VII European Congress on Computational Methods in Applied Sciences and Engineering*, pages 3843–3857, 2016.

[89] R. Cozzens. *CATIA V5 Workbook Release 19*. SDC Publications, 2009.

[90] D. Fudge, D. W. Zingg, and R. Haimes. A CAD-free and a CAD-based geometry control system for aerodynamic shape optimization. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 10 - 13 January 2005, Reno, Nevada.

[91] W. E. Brock, C. Burdyshaw, S. Karman, V. C. Betro, B. Hilbert, W. K. Anderson, and R. Haimes. Adjoint-based design optimization using CAD parameterization through CAPRI. In *proceedings of the 50th AIAA Aerospace Sciences Meeting*, 2012.

[92] A. H. Truong, D. W. Zingg, and R. Haimes. Surface mesh movement algorithm for computer-aided-design-based aerodynamic shape optimization. *AIAA Journal*, 54(2):542–556, 2016.

[93] M. Nemec and M. J. Aftosmisy. Adjoint algorithm for CAD-based shape optimization using a Cartesian method. In *17th AIAA Computational Fluid Dynamics Conferences*, Toronto, Ontario, Canada, 2005.

[94] C. G. Armstrong, T. T. Robinson, H. Ou, and C. Othmer. Linking adjoint sensitivity maps with CAD parameters. In *Evolutionary Methods for Design, Optimization and Control*, pages 234–239, 2007.

[95] T. T. Robinson, C. G. Armstrong, H. S. Chua, C. Othmer, and T. Grahs. Optimizing parameterized CAD geometries using sensitivities based on adjoint functions. *Computer-Aided Design and Applications*, 9(3):253–268, 2012.

[96] D. Agarwal, T. T. Robinson, C. G. Armstrong, S. Marques, I. Vasilopoulos, and M. Meyer. Parametric design velocity computation for CAD-based design optimization using adjoint methods. *Engineering with Computers*, pages 1–15, 2017.

[97] I. Vasilopoulos, P. Flassig, and M. Meyer. CAD-based aerodynamic optimization of a compressor stator using conventional and adjoint-driven approaches. In *ASME Turbo Expo 2017: Turbomachinery Technical Conference and Exposition*, pages V02CT47A004–V02CT47A004. American Society of Mechanical Engineers, 2017.

[98] O. C. SAS. Open cascade technology, 3D modeling & numerical simulation, 2008.

[99] J. Dannenhoffer and R. Haimes. Design sensitivity calculations directly on CAD-based geometry. In *AIAA Aerospace Sciences Meeting*, 2015.

[100] M. Banovic, O. Mykhaskiv, S. Auriemma, A. Walther, H. Legrand, and J. D. Müller. Algorithmic differentiation of the Open CASCADE Technology CAD kernel and its coupling with an adjoint CFD solver. *Optimization Methods & Software*, pages 1–16, 2017.

[101] O. Mykhaskiv, M. Banovic, S. Auriemma, P. Mohanamuraly, A. Walther, H. Legrand, , and J. Müller. NURBS-based and parametric-based shape optimisation with differentiated CAD kernel. *Computer-Aided Design and Applications*, 2018.

[102] P. M. Thompson, T. T. Robinson, and C. G. Armstrong. Efficient CAD-based aerodynamic design optimization with adjoint CFD data. In *21st AIAA Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences*, 2013.

[103] I. Vasilopoulos, D. Agarwal, M. Meyer, T. T. Robinson, and C. G. Armstrong. Linking parametric CAD with adjoint surface sensitivities. In *European Congress*

on Computational Methods in Applied Sciences and Engineering, Athens, Greece, 2016.

[104] B. Smith and J. Wellington. *Initial graphics exchange specification (IGES), version 3.0*. Society of Automotive Engineers, 1986.

[105] M. Martín, E. Andrés, M. Wildham, P. Bitrián, and C. Lozano. CAD-based aerodynamic shape design optimization with the DLR TAU code. *Proceedings of the Institution of Mechanical Engineers, Part G (Journal of Aerospace Engineering)*, pages 2010–2, 2010.

[106] L. Mueller and T. Verstraete. CAD integrated multipoint adjoint-based optimization of a turbocharger radial turbine. *International Journal of Turbomachinery, Propulsion and Power*, 2(3):14, 2017.

[107] R. Jesudasan, X. Zhang, M. Gugala, and J. D. Müller. CAD-free vs CAD-based parametrisation method in adjoint-based aerodynamic shape optimization. European Congress on Computational Methods in Applied Sciences and Engineering, Athens, Greece, 2016.

[108] J. Lépine, J.-Y. Trépanier, and F. Pépin. Wing aerodynamic design using an optimized NURBS geometrical representation. In *38th Aerospace Sciences Meeting and Exhibit*, page 669, 2000.

[109] J. Lépine, F. Guibault, J.-Y. Trépanier, and F. Pépin. Optimized nonuniform rational B-spline geometrical representation for aerodynamic design of wings. *AIAA journal*, 39(11):2033–2041, 2001.

[110] S. Painchaud-Ouellet, C. Tribes, J.-Y. Trépanier, and D. Pelletier. Airfoil shape optimization using a Nonuniform Rational B-Splines parametrization under thickness constraint. *AIAA Journal*, 44(10):2170–2178, 2006.

[111] W. A. Wall, M. A. Frenzel, and C. Cyron. Isogeometric structural shape optimization. *Computer methods in applied mechanics and engineering*, 197(33):2976–2988, 2008.

[112] X. Qian. Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 199(29-32):2059–2071, 2010.

[113] D. Fußeder and B. Simeon. Algorithmic aspects of isogeometric shape optimization. In *Isogeometric Analysis and Applications 2014*, pages 183–207. Springer, 2015.

[114] M. Ziani, R. Duvigneau, and M. Doerfel. On the role played by NURBS weights in isogeometric structural shape optimization. In *International Conference on Inverse Problems, Control and Shape Optimization*, Cartagena, Spain, April 2010.

[115] Y.-U. Song. 2D Isogeometric Shape Optimization considering both control point positions and weights as design variables. In *10th World Congress on Structural and Multidisciplinary Optimization*, 2013.

[116] T. T. Mengistu and W. S. Ghaly. A geometric representation of airfoils using NURBS. In *International Conferences on Fluid Dynamics and Propulsion*, December 18-20, 2001, Cairo, Egypt.

[117] F. J. Wessels, G. Venter, and T. Von Backström. An efficient scheme for describing airfoils using non-uniform rational b-splines. In *ASME Turbo Expo 2012: Turbine Technical Conference and Exposition*, pages 969–977. American Society of Mechanical Engineers, 2012.

[118] X. Ma, J. Ai, J. Shan, and Y. Li. Aerodynamic shape deformation of underwing nacelle-pylon configuration based on B-spline surface. In *Applied Mechanics and Materials*, volume 444, pages 191–195. Trans Tech Publ, 2014.

[119] C. Lozano, E. Andrés, M. Martín, and P. Bitrián. Domain versus boundary computation of flow sensitivities with the continuous adjoint method for aerodynamic shape optimization problems. *International Journal for Numerical Methods in Fluids*, 70(10):1305–1323, 2012.

[120] Y. Liang, X. Cheng, Z. Li, and J. Xiang. Multi-objective robust airfoil optimization based on non-uniform rational B-spline (NURBS) representation. *Science China Technological Sciences*, 53(10):2708–2717, 2010.

[121] Y. Liang, X. Cheng, Z. Li, and J. Xiang. Robust multi-objective wing design optimization via CFD approximation model. *Engineering Applications of Computational Fluid Mechanics*, 5(2):286–300, 2011.

[122] A. Yildirim and S. Eyi. Three dimensional design optimization using adjoint method. In *34th AIAA Applied Aerodynamics Conference*, page 3868, 2016.

[123] N. Leal, E. Leal, and J. W. Branch. Simple method for constructing NURBS surfaces from unorganized points. In *Proceedings of the 19th international meshing roundtable*, pages 161–175. Springer, 2010.

[124] M. Martin, E. Andres, E. Valero, and C. Lozano. Gradients calculation for arbitrary parameterizations via volumetric NURBS: the control box approach. In *EUCASS conference*, 2013.

[125] S. Xu, S. Timme, O. Mykhaskiv, and J. D. Müller. Wing-body junction optimisation with CAD-based parametrisation including a moving intersection. *Aerospace Science & Technology*, 68, 2017.

[126] D. E. Goldberg. *Genetic algorithms*. Pearson Education India, 2006.

[127] T. Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[128] O. Chernukhin and D. W. Zingg. Multimodality and global optimization in aerodynamic design. *AIAA journal*, 2013.

[129] G. M. Streuber and D. W. Zingg. Investigation of multimodality in aerodynamic shape optimization based on the Reynolds-Averaged Navier-Stokes equations. In *35th AIAA Applied Aerodynamics Conference*, page 3752, 2017.

[130] N. P. Bons, X. He, C. A. Mader, and J. R. Martins. Multimodality in aerodynamic wing design optimization. In *35th AIAA Applied Aerodynamics Conference*, page 3753, 2017.

[131] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[132] M. Bartholomew-Biggs. *Nonlinear optimization with engineering applications*, volume 19. Springer, 2008.

[133] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[134] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[135] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[136] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.

[137] J. L. Morales and J. Nocedal. Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Transactions on Mathematical Software (TOMS)*, 38(1):7, 2011.

[138] P. Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

[139] J.-D. Müller. A review of adjoint method, Queen Mary University of London. 2007.

[140] W. K. Anderson, J. C. Newman, D. L. Whitfield, and E. J. Nielsen. Sensitivity analysis for Navier-Stokes equations on unstructured meshes using complex variables. *AIAA journal*, 39(1):56–63, 2001.

[141] J. R. R. A. Martins, I. Kroo, and J. J. Alonso. An automated method for sensitivity analysis using complex variables. In *38th Aerospace Sciences Meeting and Exhibit*, page 689, 2000.

[142] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The connection between the complex-step derivative approximation and algorithmic differentiation. In *39th Aerospace Sciences Meeting and Exhibit*, page 921, 2001.

[143] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.

[144] K.-L. Lai and J. Crassidis. Generalizations of the complex-step derivative approximation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6348, 2006.

[145] J.-D. Müller. Lecture notes on numerical optimisation, Queen Mary University of London. February 2014.

[146] J.-D. Müller and P. Cusdin. On the performance of discrete adjoint CFD codes using automatic differentiation. *International journal for numerical methods in fluids*, 47(8-9):939–945, 2005.

[147] A. G. Baydin and B. A. Pearlmutter. Automatic differentiation of algorithms for machine learning. *arXiv preprint arXiv:1404.7456*, 2014.

[148] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767*, 2015.

[149] F. Srajer, Z. Kukelova, A. Fitzgibbon, A. Schwing, M. Pollefeys, and T. Pajdla. A benchmark of selected algorithmic differentiation tools on some problems in machine learning and computer vision. *Automatic Differentiation*, 2016.

[150] L. Hascoët and V. Pascual. Tapenade 2.1 user's guide. Technical report, INRIA, 2004.

[151] L. B. Rall. Automatic differentiation: Techniques and applications. 1981.

[152] G. Corliss. *Automatic differentiation of algorithms: from simulation to optimization*, volume 1. Springer, 2002.

[153] A. Griewank. A mathematical view of automatic differentiation. *Acta Numerica*, 12:321–398, 2003.

[154] M. Giles, D. Ghate, and M. Duta. Using automatic differentiation for adjoint CFD code development. 2005.

[155] H. M. Bücker, G. Corliss, P. Hovland, U. Naumann, and B. Norris. *Automatic differentiation: applications, theory, and implementations*, volume 50. Springer Science & Business Media, 2006.

[156] C. H. Bischof, H. M. Bücker, P. Hovland, U. Naumann, and J. Utke. Advances in automatic differentiation. 2008.

[157] S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther. *Recent advances in algorithmic differentiation*, volume 87. Springer Science & Business Media, 2012.

[158] M. Spiegel and S. Lipschutz. *Schaum's Outline of Vector Analysis*. McGraw Hill Professional, 2nd edition, 2009.

[159] D. John and J. Anderson. *Computational fluid dynamics: the basics with applications*. McGraw-Hill Companies, 1995.

[160] J. Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[161] W. S. Brainerd, C. Goldberg, and J. Adams. *Programmer's guide to Fortran 90*. Intertext Publications, Inc.,/McGraw-Hill, Inc., 1990.

[162] S. J. Chapman. *Fortran 90/95 for scientists and engineers*. McGraw-Hill Higher Education, 2004.

[163] H. A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2):631–644, 1992.

[164] F. J. Blom. Considerations on the spring analogy. *International journal for numerical methods in fluids*, 32(6):647–668, 2000.

[165] C. Degand and C. Farhat. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers & structures*, 80(3):305–316, 2002.

[166] M. Selim and R. Koomullil. Mesh deformation approaches - a survey. *Journal of Physical Mathematics*, 7(181):2090–0902, 2016.

[167] Y. Wang. *Roubust and stable discrete adjoint solver development for shape optimisation of incompressible flows with industrial applications*. PhD thesis, Queen Mary University of London, 2016.

[168] U. Ghia, K. N. Ghia, and C. T. Shin. High-resolutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411, 1982.

[169] Y. Wang, S. Akbarzadeh, and J.-D. Müller. Stabilisation of discrete adjoint solvers for incompressible flow. In *22nd AIAA Computational Fluid Dynamics Conference*, page 2749, 2015.

[170] J.-D. Müller, M. Gugala, S. Xu, J. Hüchelheim, P. Mohanamuraly, and O. R. Imam-Lawal. Introducing STAMPS: an open-source discrete adjoint CFD solver using source-transformation AD. In *11th ASMO UK/ISSMO/NOED2016: International Conference on Nuemerical Optimisation Methods for Enggineering Design*, 2016.

[171] P. R. Spalart and C. L. Rumsey. Effective inflow conditions for turbulence models in aerodynamic calculations. *AIAA journal*, 45(10):2544, 2007.

[172] S. R. Allmaras and F. T. Johnson. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. In *Seventh international conference on computational fluid dynamics (ICCFD7)*, pages 1–11, 2012.

[173] P. L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2):357–372, 1981.

[174] S. Xu, D. Radford, M. Meyer, and J.-D. Müller. Stabilisation of discrete steady adjoint solvers. *Journal of Computational Physics*, 299:175–195, 2015.

[175] A. Jaworski, P. Cusdin, and J. Müller. Uniformly converging simultaneous time-stepping methods for optimal design. *Eurogen, Munich*, 2005.

[176] S. Hazra, V. Schulz, J. Brezillon, and N. Gauger. Aerodynamic shape optimization using simultaneous pseudo-timestepping. *Journal of Computational Physics*, 204(1):46–64, 2005.

[177] J. A. Witteveen and H. Bijl. Explicit mesh deformation using inverse distance weighting interpolation. In *19th AIAA Computational Fluid Dynamics Conference, AIAA, San Antonio, Texas*, 2009.

[178] S. Xu. *CAD-based CFD shape optimisation using discrete adjoint solvers*. PhD thesis, Queen Mary University of London, 2015.

[179] V. Schmitt and F. Charpin. Pressure distributions on the ONERA-M6-Wing at transonic Mach numbers. *Experimental data base for computer program assessment*, 4, 1979.

[180] M. Gugala. *Output-based mesh adaptation using geometric multi-grid for error estimation*. PhD thesis, Queen Mary University of London, 2017.

[181] J. Lions. *Optimal control of systems governed by partial differential equations*. Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1971.

[182] M. B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65(3-4):393–415, 2000.

[183] T. Leblond, P. Froment, P. De Nazelle, R. Sellakh, P. Serré, and G. Chevallier. Gradient-based optimization of parameterized CAD geometries. In *11th World Congress on Structural and Multidisciplinary Optimization*, Sydney, Australia, June 2015. ISSMO.

[184] J. J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimlinger, and D. Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1. *Journal of Aircraft*, 36(1):51–60, 1999.

[185] W. K. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids*, 28(4):443–480, 1999.

[186] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. In *34th Aerospace Sciences Meeting and Exhibit*, page 94, Reno, Nevada, 1996.

[187] D. Wang and L. He. Adjoint aerodynamic design optimization for blades in multistage turbomachines-part I: Methodology and verification. *Journal of Turbomachinery*, 132(2):021011, 2010.

[188] S. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *AIAA paper*, 667:2000, 2000.

[189] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. SU2: an open-source suite for multiphysics simulation and design. *AIAA Journal*, 2015.

[190] D. J. Mavriplis. Discrete adjoint-based approach for optimization problems on three-dimensional unstructured meshes. *AIAA journal*, 45(4):741–750, 2007.

[191] Y. Gao, Y. Wu, and J. Xia. Automatic differentiation based discrete adjoint method for aerodynamic design optimization on unstructured meshes. *Chinese Journal of Aeronautics*, 30(2):611–627, 2017.

[192] J. D. Mueller, S. Xu, D. Jones, F. Christakopoulos, and W. Jahn. Development and application of discrete adjoint CFD codes using automatic differentiation. (ECCOMAS):1–17, 2012.

[193] G. Lombardi and M. Maganzi. Use of the CFD for the aerodynamic optimization of the car shape: Problems and application. (July), 2009.

[194] C. Comis Da Ronco, R. Ponza, and E. Benini. Aerodynamic shape optimization of aircraft components using an advanced multi-objective evolutionary approach. *Computer Methods in Applied Mechanics and Engineering*, 285:255–290, 2015.

[195] J. Zaya. Aerodynamic optimization of ground vehicles with the use of Fluent's adjoint solver. Master's thesis, Chalmers University of Technology, 2013.

[196] A. Tzanakis. Duct optimization using CFD software 'ANSYS Fluent adjoint solver'. Master's thesis, Chalmers University of Technology, 2014.

[197] T. Leblond, G. Chevallier, P. D. Nazelle, and P. Froment. CAD-based shape optimization of an exhaust manifold using the adjoint solver. In *STAR Global Conference 2016*, Prague, Czech Republic. 2016.

[198] C. Othmer. A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *International Journal for Numerical Methods in Fluids*, 58(8):861–877, 2008.

[199] F. Palacios, M. R. Colonno, A. C. Aranake, A. Campos, S. R. Copeland, T. D. Economon, A. K. Lonkar, T. W. Lukaczyk, T. W. R. Taylor, and J. J. Alonso. Stanford university unstructured (SU 2): An open-source integrated computational environment for multi-physics simulation and design. *Aiaa Journal*, 2013:1–60, 2013.

[200] F. Palacios, T. D. Economon, A. Aranake, S. R. Copeland, A. K. Lonkar, T. W. Lukaczyk, D. E. Manosalvas, K. R. Naik, S. Padron, and B. Tracey. Stanford university unstructured (SU2): Analysis and design technology for turbulent flows. In *Aerospace Sciences Meeting*, 2014.

[201] T. D. Economon, F. Palacios, J. J. Alonso, G. Bansal, D. Mudigere, A. Deshpande, A. Heinecke, and M. Smelyanskiy. Towards high-performance optimizations of the unstructured open-source SU2 suite. In *AIAA Infotech @ Aerospace*, 2015.

[202] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. SU2: An open-source suite for multiphysics simulation and design. *Aiaa Journal*, 54(3):1–19, 2016.

[203] C. Othmer, E. M. Papoutsis-kiachagias, and K. Haliskos. CFD optimization via sensitivity-based shape morphing. In *4th ANSA and uETA International Conference*, pages 2–9, 2010.

[204] T. Verstraete and J. Li. Multi-objective optimization of a U-bend for minimal pressure loss and maximal heat transfer performance in internal cooling channels. In *ASME Turbo Expo 2013: Turbine Technical Conference and Exposition*, 2013.

[205] G. Yang and A. D. Ronch. Aerodynamic shape optimisation of benchmark problems using SU2. In *AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018.

[206] Y. Yang and S. Özgen. Implementation of ball-center spring analogy mesh deformation technique with CFD design optimization. In *Aiaa Computational Fluid Dynamics Conference*, 2013.

[207] A. Vuruskan and S. Hosder. Investigation of the effectiveness of shape parameterization techniques for robust aerodynamic optimization. In *Aiaa Applied Aerodynamics Conference*, 2017.

[208] A. Amrit, X. Du, A. S. Thelen, L. T. Leifsson, and S. Koziel. Aerodynamic design of the RAE 2822 in transonic viscous flow: Single- and multi-objective optimization studies. In *AIAA Aviation Forum 5-9 June 2017, Denver, Colorado AIAA Applied Aerodynamics Conference*, 2017.

[209] F. Palacios, T. D. Economon, A. D. Wendorff, and J. J. Alonso. Large-scale aircraft design using SU2. In *53rd AIAA Aerospace Sciences Meeting*, 2015.

[210] Y. L. Ma and W. T. Hewitt. Point inversion and projection for NURBS curve and surface: control polygon approach. *Computer Aided Geometric Design*, 20(2):79–99, 2003.

[211] X.-D. Chen, H. Su, J.-H. Yong, J.-C. Paul, and J.-G. Sun. A counterexample on point inversion and projection for NURBS curve. *Computer Aided Geometric Design*, 24(5):302, 2007.

[212] C. K. Au and M. M. F. Yuen. Unified approach to NURBS curve shape modification. *Computer-Aided Design*, 27(2):85–93, 1995.

[213] G. E. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002.

[214] J. Xu, W. Liu, J. Wu, H. Bian, and L. Li. Geometric algorithm for point projection and inversion onto Bézier surfaces. *Frontiers of Computer Science in China*, 3(4):472–476, 2009.

[215] S. M. Hu and J. Wallner. A second order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design*, 22(3):251–260, 2005.

[216] M. R. Hidalgo. Improvement of point inversion algorithm for NURBS curves and surfaces. In *WCCM8/ECCOMAS 2008*, June 30 - July 5, 2008, Venice, Italy.

[217] P. H. Calamai and J. J. Moré. Projected gradient methods for linearly constrained problems. *Mathematical Programming*, 39(1):93–116, 1987.

[218] H. Andrews and C. Patterson. Singular value decompositions and digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(1):26–53, 1976.

[219] C.-C. Lai and C.-C. Tsai. Digital image watermarking using discrete wavelet transform and singular value decomposition. *IEEE Transactions on instrumentation and measurement*, 59(11):3060–3063, 2010.

[220] E. Biglieri and K. Yao. Some properties of singular value decomposition and their applications to digital signal processing. *Signal Processing*, 18(3):277–289, 1989.

[221] P. P. Kanjilal, S. Palit, and G. Saha. Fetal ECG extraction from single-channel maternal ECG using singular value decomposition. *IEEE Transactions on Biomedical Engineering*, 44(1):51–59, 1997.

[222] A. A. Maciejewski and C. A. Klein. The singular value decomposition: Computation and applications to robotics. *The International journal of robotics research*, 8(6):63–79, 1989.

[223] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

[224] V. Klema and A. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

[225] L. V. Foster and T. A. Davis. Algorithm 933: Reliable calculation of numerical rank, null space bases, pseudoinverse solutions, and basic solutions using suitesparseqr. *ACM Transactions on Mathematical Software (TOMS)*, 40(1):7, 2013.

[226] G. Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, 2011.

[227] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammerling, J. Demmel, C. Bischof, and D. Sorensen. LAPACK: A portable linear algebra library for high-performance computers. In *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, pages 2–11, 1990.

[228] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' guide*, volume 9. Siam, 1999.

[229] L. Hogben. *Handbook of linear algebra*. CRC Press, 2006.

[230] G. Strang. *Linear Algebra and Its Applications*. Thomson, Brooks/Cole, 2006.

[231] T. M.-M. Leung. *A Newton-Krylov approach to aerodynamic shape optimization in three dimensions*. PhD thesis, University of Toronto, 2010.

[232] T. M. Leung and D. W. Zingg. Aerodynamic shape optimization of wings using a parallel newton-krylov approach. *AIAA Journal*, 50(3):540–550, 2012.

[233] M. J. Pratt. Introduction to ISO 10303 - the STEP standard for product data exchange. *Journal of Computing and Information Science in Engineering*, 1(1):102, 2001.

[234] D. Loffredo. Fundamentals of STEP implementation. *STEP Tools, Inc*, pages 1–12, 1999.

[235] A. B. Feeney and D. M. Price. A modular architecture for STEP. In *Proceedings of world automation congress*, 2000.

[236] M. P. Bhandarkar, B. Downie, M. Hardwick, and R. Nagi. Migrating from IGES to STEP: One to one translation of IGES drawing to STEP drafting data. *Computers in Industry*, 41(3):261–277, 2000.

[237] L. Ma, Z. He, and J. Dong. Advanced curve and surface design in CAD system using STEP. In *IEEE Region 10 Conference on Computer, Communication, Control and Power Engineering*, volume 1, pages 581–584, Oct 1993.

[238] S. Sivakumar and V. Dhanalakshmi. A feature-based system for CAD/CAM integration through STEP file for cylindrical parts. *Indian Journal of Engineering and Materials Sciences*, 20(1):21–26, 2013.

[239] A. Iyer, S. G. Kapoor, and R. E. DeVor. CAD data visualization for machining simulation using the STEP standard. *Journal of Manufacturing Systems*, 20(3):198–209, 2001.

[240] C. R. Gilman and S. J. Rock. The use of STEP to integrate design and solid freeform fabrication. In *Proceedings of Solid Freeform Fabrication Symposium*, 1995.

[241] B. Starly, A. Lau, W. Sun, W. Lau, and T. Bradbury. Direct slicing of STEP based NURBS models for layered manufacturing. *Computer-Aided Design*, 37(4):387–397, 2005.

[242] J. Kim, M. J. Pratt, R. Iyer, and R. Sriram. Data exchange of parametric cad models using ISO 10303-108. *NIST intergovernmental report*, 2007.

[243] B. C. Kim, D. Mun, S. Han, and M. J. Pratt. A method to exchange procedurally represented 2D CAD model data using ISO 10303 STEP. *Computer-Aided Design*, 43(12):1717–1728, 2011.

[244] M. Bhandarkar and R. Nagi. STEP-based feature extraction from STEP geometry for agile manufacturing. *Computers in Industry*, 2000.

[245] W. Xiao, Y. Liu, R. Li, W. Wang, J. Zheng, and G. Zhao. Reconsideration of T-spline data models and their exchanges using STEP. *Computer-Aided Design*, 79:36–47, 2016.

[246] J. Procházková and D. Procházka. Implementation of NURBS curve derivatives in engineering practice. *WSCG 2007 Poster Papers Proceedings*, 0(2):5–8, 2007.

[247] M. Martin, E. Valero, C. Lozano, and E. Andres. Treatment of surface intersections for gradient-based aerodynamic shape optimization. In *5th EUCASS*, 2013.

[248] M. Martin, M. Cordero-Gracia, and M. Gómez. Adaptation of the computational grid to a moving wing-fusselage intersection via NURBS and radial basis. *wccm-eccm-ecfd2014.org*, 2014.

[249] M. Martin, E. Valero, C. Lozano, and E. Andres. Gradient calculation for arbitrary parameterizations via volumetric NURBS: The control box approach. 2013.

[250] M. Gugala, S. Xu, and J.-D. Müller. Node-based VS CAD-based approach in CFD adjoint-based shape optimisation. In *6th European Conference on Computational Fluid Dynamics*, July 20-25, 2014, Barcelona, Spain.

[251] A. Taherkhani, P. Gaskell, R. Hewson, Z. Khatir, A. Polynkin, H. Thompson, and V. Toropov. Aerodynamic design optimization of emergency response vehicle using computational fluid dynamics. In *Proceedings of the 9th ASMO UK Conference Engineering Design Optimisation.*

[252] S. Berger, L. Talbot, and L. Yao. Flow in curved pipes. *Annual review of fluid mechanics*, 15(1):461–512, 1983.

[253] D. Jones, J.-D. Mueller, and S. Bayyuk. CFD development with automatic differentiation. *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, pages 1–11, 2012.

[254] S. Akbarzadeh, Y. Wang, and J. D. Mueller. Fixed point discrete adjoint of simple-like solvers. In *22nd AIAA Computational Fluid Dynamics Conference*, 22-26 June 2015, Dallas, TX.

[255] J.-D. Müller, D. Jones, W. Jahn, C. Othmer, M. Megahed, N. Hollette, G. Pierrot, K.-U. Bletzinger, and E. Stavropolou. Goal-based flow optimisation for automotive design. *Procedia-Social and Behavioral Sciences*, 48:3599–3612, 2012.

[256] H. Namgoong, P. Ireland, and C. Son. Optimisation of a 180° U-shaped bend shape for a turbine blade cooling passage leading to a pressure loss coefficient of approximately 0.6. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 230(8):1371–1384, June 2016.

[257] T. Verstraete. *Multidisciplinary turbomachinery component optimization considering performance, stress, and internal heat transfer.* PhD thesis, Von Karman Institute, 2008.

[258] M. B. Akin and W. Sanz. A quasi-first order optimization method and its demonstration on the optimization of a U-bend. In *ASME Turbo Expo 2015: Turbine Technical Conference and Exposition.*

[259] K. T. Tsiakas, X. S. Trompoukis, V. G. Asouti, M. S. G. Nejad, and K. C. Giannakoglou. Shape optimization on GPUs, using the continuous adjoint method and volumetric NURBS. In *8th GRACM International Congress on Computational Mechanics*, number July, 2015.

[260] R. Jesudasan, X. Zhang, and J.-D. Müller. Adjoint optimisation of internal turbine cooling channel using NURBS-based automatic and adaptive parametrisation method. In *Proceedings of the fifth ASME Gas Turbine India Conference, December 7-8*, 2017.

[261] T.-M. Liou, Y.-Y. Tzeng, and C.-C. Chen. Fluid flow in a 180 deg sharp turning duct with different divider thicknesses. *ASME J. Turbomach*, 121(3):569–576, 1999.

[262] Z. Lyu, G. K. Kenway, C. Paige, and J. Martins. Automatic differentiation adjoint of the Reynolds-averaged Navier-Stokes equations with a turbulence model. In *21st AIAA computational fluid dynamics conference, San Diego, CA*, volume 10, pages 6–2013, 2013.

[263] S. Schmidt, C. Ilic, V. Schulz, and N. R. Gauger. Three-dimensional large-scale aerodynamic shape optimization based on shape calculus. *AIAA journal*, 2013.

[264] D. A. Masters, N. J. Taylor, T. Rendall, and C. B. Allen. Three-dimensional subdivision parameterisation for aerodynamic shape optimisation. In *55th AIAA Aerospace Sciences Meeting*, page 0036, 2017.

[265] D. Agarwal, S. Marques, T. T. Robinson, C. G. Armstrong, and P. Hewitt. Aerodynamic shape optimization using feature based CAD systems and adjoint methods. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 3999, 2017.

[266] S. Painchaud-Ouellet, C. Tribes, J.-Y. Trepanier, and D. Pelletier. Airfoil shape optimization using NURBS representation under thickness constraint. In *42nd AIAA aerospace sciences meeting and exhibit*, volume 1095. Reno, Nevada, 2004.

[267] W. Miller. The formula for curvature. 2007.

[268] S. Akbarzadeh, X. Zhang, and J.-D. Müller. On the fixed-point iteration discrete adjoint gradient of SIMPLE-type incompressible flow solvers. *International Journal for Numerical Methods in Fluids*, submitted.

[269] X. Zhang, Y. Wang, M. Gugala, and J.-D. Müller. Geometric continuity constraints for adjacent NURBS patches in shape optimisation. In *proceedings of VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress)*, 2016.

[270] X. Zhang, R. Jesudasan, and J.-D. Müller. Adjoint-based aerodynamic optimisation of wing shape using Non-Uniform Rational B-splines. In *International Conference On Evolutionary and Deterministic Methods For Design Optimisation And Control With Applications to Industrial And Societal Problems (Eurogen 2017)*, September 13 - 15, 2017, Madrid, Spain.

[271] X. Han and D. W. Zingg. An adaptive geometry parametrization for aerodynamic shape optimization. *Optimization and Engineering*, pages 1–23, 2013.

[272] C. Severance. Discovering javascript object notation. *Computer*, 45(4):6–8, 2012.

[273] T. Bray. The javascript object notation (JSON) data interchange format, RFC 7159, RFC Editor, March 2014.